SNAP PAC MOTION CONTROL USER'S GUIDE

SNAP-SCM-MCH16 Motion Module
SNAP-SCM-BB4 Breakout Board
OptoMotion Command Set

Form 1673-070307—March, 2007

OPTO 22

43044 Business Park Drive • Temecula • CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
Email: support@opto22.com
Web: support.opto22.com

SNAP PAC Motion Control User's Guide
Form 1673-070307—March, 2007

Copyright © 2003–2006 Opto 22.
All rights reserved.
Printed in the United States of America.

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or later are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. These products include, but are not limited to, OptoTerminal-G70, OptoTerminal-G75, and Sony Ericsson GT-48; see the product data sheet for specific warranty information. Refer to Opto 22 form number 1042 for complete warranty information.

Cyrano, Opto 22 FactoryFloor, Optomux, and Pamux are registered trademarks of Opto 22. Generation 4, ioControl, ioDisplay, ioManager, ioProject, ioUtilities, mistic, Nvio, Nvio.net Web Portal, OptoConnect, OptoControl, OptoDataLink, OptoDisplay, OptoOPCServer, OptoScript, OptoServer, OptoTerminal, OptoUtilities, PAC Control, PAC Display, PAC Manager, PAC Project, SNAP Ethernet I/O, SNAP I/O, SNAP OEM I/O, SNAP PAC System, SNAP Simple I/O, SNAP Ultimate I/O, and SNAP Wireless LAN I/O are trademarks of Opto 22.

ActiveX, JScript, Microsoft, MS-DOS, VBScript, Visual Basic, Visual C++, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Unicenter is a registered trademark of Computer Associates International, Inc. ARCNET is a registered trademark of Datapoint Corporation. Modbus is a registered trademark of Schneider Electric. Wiegand is a registered trademark of Sensor Engineering Corporation. Nokia, Nokia M2M Platform, Nokia M2M Gateway Software, and Nokia 31 GSM Connectivity Terminal are trademarks or registered trademarks of Nokia Corporation. Sony is a trademark of Sony Corporation. Ericsson is a trademark of Telefonaktiebolaget LM Ericsson.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Table of Contents

Chapter 1: Overview 5

Introduction 5

Software 6

Software Availability 6

About this Guide 6

Related Documentation 7

For Help 7

Compatibility 7

Chapter 2: Quick Start 9

What You Will Need 9

Setting Up the SNAP-SCM-MCH16 Module 10

Configuring the Breakout Board 12

Assigning an IP Address to the Controller or Brain 13

Checking the Firmware Version of the Controller or Brain 16

Using an Example Strategy 16

Removing a Module 17

Chapter 3: Specifications 19

Module Specifications 20

Module Bias and Termination 20

Module LEDs 20

Breakout Board Specifications 21

Breakout Board LEDs 21

Power and Serial LEDs 21
Axis LEDs 21
Breakout Board Connector Pins 22
J2: Serial Connector 22
J3 (and J6, J9, & J12): Encoder Signal Inputs 22
J4 (and J7, J10, & J13): Stepper Motor Outputs 22
J5 (and J8, J11, & J14): Stepper Motor Inputs 22
J15: Auxiliary Power Input 22
Breakout Board Switches 23
SW1–SW8: Signal Selection for Encoder Inputs 23
S3 (and S5, S4, & S6): Enable/Disable Axis 23
S7: Bias & Termination, Voltage Select, Breakout Board Address 23
S1 & S2: Pullup Resistors 23
RS-422/485 Serial Cable. 24
Component Connection Schematic 25
Connecting Multiple Breakout Boards 26
Calculating Power Requirements 27

Chapter 4: Using Motion Control Commands 29
Using the Commands 29
Entering Commands in OptoScript 30
Host I/O Errors 31

Chapter 5: Command Reference 33
Commands Listed by Group 33
Command Details 35
Commands in Alphabetical Order 37

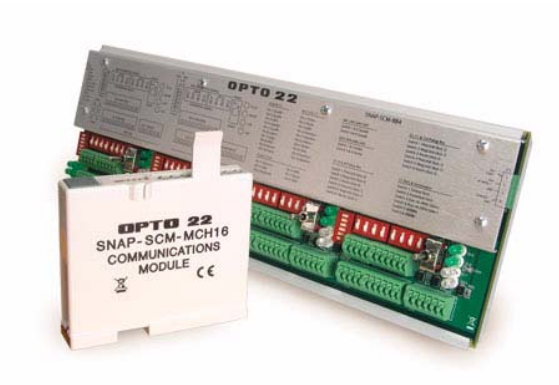
Appendix A: SNAP-SCM-MCH16 Conversion Formulas 113

Overview

Introduction

The easy-to-use SNAP PAC Motion Control Subsystem provides an integrated hardware and software toolset for controlling multi-axis stepper motors. The subsystem consists of:

- SNAP Motion Control host communication modules (SNAP-SCM-MCH16)
- SNAP Motion Control breakout boards (SNAP-SCM-BB4)
- OptoMotion command set, a free library of motion commands.



The **SNAP-SCM-MCH16** motion control host module is a serial communication module that links up to four SNAP-SCM-BB4 motion control breakout boards with a SNAP PAC I/O unit. When mounted on an I/O unit and connected to a breakout board, a single SNAP-SCM-MCH16 module allows a SNAP PAC controller running a PAC Control™ programming strategy to control up to 16 stepper motors. The module snaps into an Opto 22 SNAP PAC mounting rack right beside digital and analog modules. LED indicators are provided to indicate Transmit and Receive on each port.

Each **SNAP-SCM-BB4** breakout board is equipped with a Magellan™ processor chip set that outputs pulse and direction signals for up to four stepper motor systems. You can daisy-chain up to four breakout boards connected to a single module. The module's external connector provides lines to power one breakout board; additional boards require a separate power source. The SNAP-SCM-BB4 breakout board is designed to be mounted using a DIN-rail system.

The **OptoMotion** library of motion commands supports many of the Magellan™ Motion Processor commands. These commands are entered in a PAC Control strategy as text strings using the Transmit String and Receive commands or the TransmitReceiveString command in OptoScript. The OptoMotion commands give you the ability to define and acquire motion process data such as position, velocity, acceleration, breakpoints, interrupts, and time intervals. In addition, you can execute motion-related actions such as smooth stops, stepping, and position adjustments.

Software

SNAP PAC controllers use Opto 22's **PAC Project** Microsoft® Windows®-compatible automation software for programming, human-machine-interface (HMI) development, and OPC connectivity. Two versions of PAC Project are available:

- **PAC Project Basic** includes PAC Control for developing control programs, PAC Display™ for creating operator interfaces, and PAC Manager™ configuration software.
- **PAC Project Professional** adds expanded versions of PAC Control and PAC Display plus OptoOPCServer™ software for exchanging data with OPC 2.0-compliant client software applications.

For more information, see the *PAC Project Data Sheet*.

Software Availability

PAC Project Basic is included with SNAP PAC controllers and is a free download from the Opto 22 website. PAC Project Professional is available for purchase on a CD with both Acrobat PDF format and printed documentation.

To get PAC Project Professional immediately, you can buy and download the software from the Opto 22 website at www.opto22.com; the CD and printed documentation will be shipped to you. You can also separately purchase PAC Control Professional, PAC Display Professional, and OptoOPCServer as needed. For additional information, see the *PAC Project data sheet*, Opto 22 form #1699.

About this Guide

This guide shows you how to install and use the SNAP PAC Motion Control Subsystem. This guide assumes that you know how to create a strategy in PAC Control, and how to use motion control technology. If you are not familiar with these subjects, we strongly suggest you consult commercially available resources to learn about them before attempting to install or use the SNAP PAC Motion Control Subsystem.

The following sections are included in this user's guide:

Chapter 1, "Overview"—Information about the guide and how to reach Opto 22 Product Support.

Chapter 2, "Quick Start"—Quick-start steps to get SNAP PAC Motion Control Subsystem up and running quickly.

Chapter 5, "Command Reference"—Conceptual information on networking and communicating with SNAP PAC R-series controllers; descriptions of features, capabilities, and specifications for SNAP PAC R-series controllers.

Related Documentation

See the following documents for additional information.

For this information	See this guide	Form #
Designing flowchart-based control programs for the system	*PAC Control User's Guide*	1700
	PAC Control Command Reference	1701
Configuring I/O points and system functions	*PAC Manager User's Guide*	1704
Installing and using SNAP PAC brains and I/O units.	*SNAP PAC Brains User's Guide*	1690
Installing and using Opto 22's SNAP PAC R-series of programmable automation controllers	*SNAP PAC R-Series Controllers User's Guide*	1595

For Help

If you have problems installing or using SNAP PAC Motion Control Subsystem, first check this guide and the Troubleshooting section of the user's guide for your Opto 22 hardware. If you cannot find the help you need in the guides or on the Opto 22 website, contact Opto 22 Product Support.

Phone: 800-TEK-OPTO (835-6786)
951-695-3080
(Hours are Monday through Friday, 7 a.m. to 5 p.m. Pacific Time)

Fax: 951-695-3017

Email: support@opto22.com

Opto 22 website: www.opto22.com

NOTE: Email messages and phone calls to Opto 22 Product Support are grouped together and answered in the order received.

Compatibility

SNAP-SCM-MCH16 motion modules are designed to work with SNAP PAC mounting racks. For information on using older SNAP racks, see form #1688, *SNAP PAC System Migration Tech Note*.

Quick Start

This chapter describes how to set up the Motion Control Subsystem hardware and how to run some example strategies to test your setup.

In This Chapter

What You Will Need 9
Setting Up the SNAP-SCM-MCH16 Module.....10
Configuring the Breakout Board.....12
Using an Example Strategy 16
Removing a Module 17

What You Will Need

To set up the Motion Control Subsystem you need the following things:

- SNAP-SCM-MCH16 motion module
- SNAP-SCM-BB4 motion control breakout board
- RS-422/485 serial cable (see "RS-422/485 Serial Cable" on page 24)
- PC running Opto 22 PAC Project software version 8.0 or newer
- SNAP PAC rack-mounted controller or brain[1] with firmware version 8.0 or better
- A SNAP PAC rack[2]
- Power supply[2]
- Stepper motor

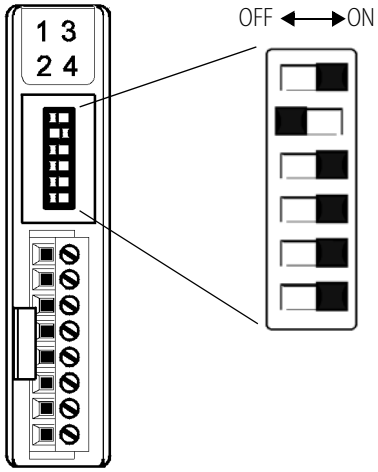
[1] I/O unit must be part of a system using PAC Control on a SNAP PAC controller.

[2] Assemble the hardware according to the directions that came with it. For help with wiring, see product data sheets, which are available on our website at www.opto22.com.

Setting Up the SNAP-SCM-MCH16 Module

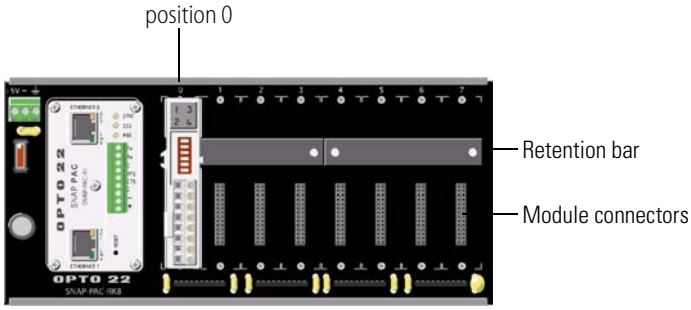
The SNAP-SCM-MCH16 module snaps into place in the row of connectors on any SNAP PAC rack. Each module connector has a number.

1. Set the termination and bias for the SNAP-SCM-MCH16 as follows:



2. Place the rack so that the module connector numbers are right-side up, with zero on the left.
3. With the power off, position the SNAP-SCM-MCH16 module over the module connector in position 0, aligning the small slot at the base of the module with the retention bar on the rack.

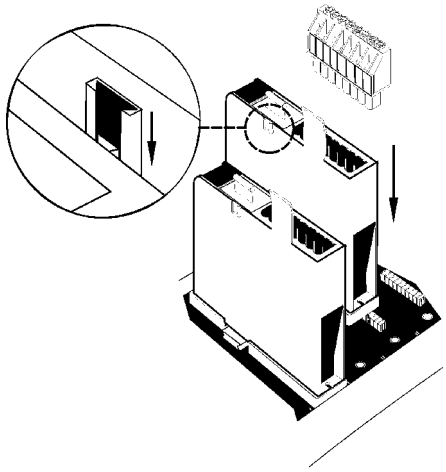
   The module can be placed in any position on the rack. This example shows the module in position 0.



4. With the module correctly aligned over the connector, push on the module to snap it into place.

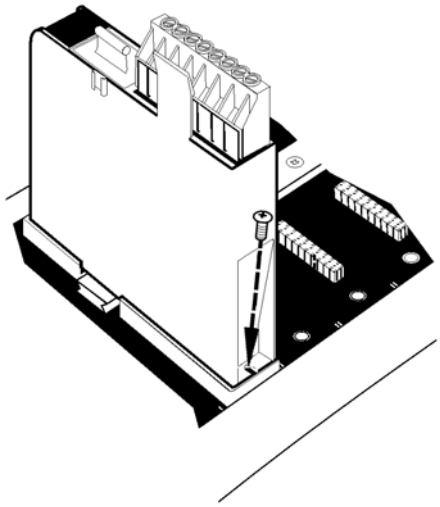
   When positioning modules next to each other, be sure to align the male and female module keys (shown in the detailed view in the illustration at right) before snapping a module into position.

   Modules snap securely into place and require a special tool (provided) to remove them. To remove a module, see page 17.



5. (Optional) As shown in the photo at right, use standard 4-40 x 1/4 truss-head Phillips hold-down screws to secure both sides of each module.

   CAUTION: *Do not over-tighten screws.*



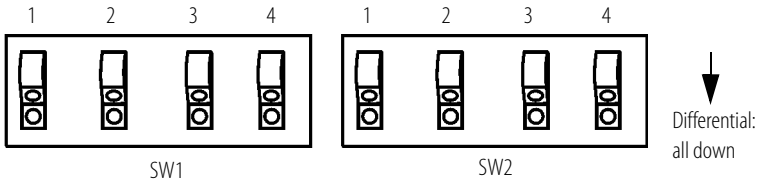
6. Make sure the rack is wired to receive 5.0 to 5.2 VDC @ 4A. Leave the power supply off at this time. For more information on power requirements, see , the SNAP I/O Racks Data Sheet.

Configuring the Breakout Board

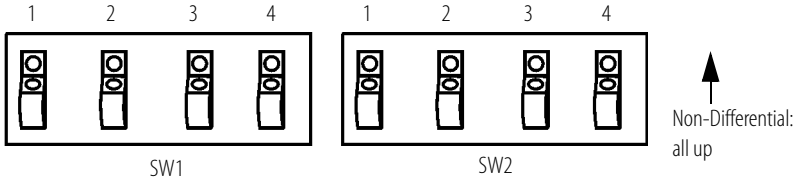
For information on breakout board specifications and the placement of switches, see "Breakout Board Specifications" on page 21.

1. Set the switches on the SNAP-SCM-BB4, making sure SW1–SW8 are in the correct position for your encoder (if you have one).

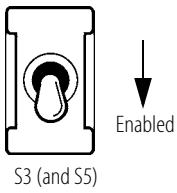
   If your encoder outputs differential signals, make sure they are in the Differential Signals position: all *down*.



   If your encoder outputs Non-Differential Signals make sure they are in the Non-Differential Signals position: all *up*.

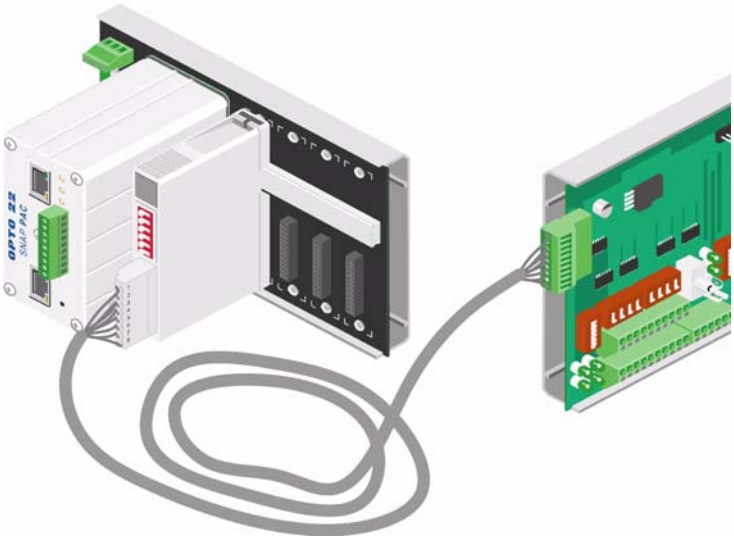


2. Enable Axis 0 by setting S3 to the Enabled position. If you are using the Electronic Gearing example strategy (see page 16), enable Axis 1 too by setting S5 to the Enabled position.

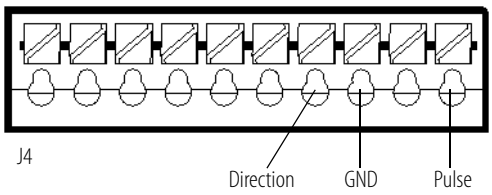


3. Set the SNAP-SCM-BB4 address and termination on S7 as shown at right.

4. Connect the SNAP-SCM-MCH16 to the SNAP-SCM-BB4 with the RS-422/485 serial cable (see "RS-422/485 Serial Cable" on page 24).



5. Connect your stepper motors to the SNAP-SCM-BB4. Minimum connections are Pulse, Direction, and GND. Make sure you have a power supply for the stepper motor.



6. If the controller or brain on the rack already has an IP address, skip to "Checking the Firmware Version of the Controller or Brain" on page 16. If the controller or brain does not have an IP address (indicated by the STAT LED blinking orange), to assign an IP address to the controller, following steps in the next section, before using the controller.

Assigning an IP Address to the Controller or Brain

The following instructions for assigning an IP address are provided here in simplified form for your convenience. For detailed instructions on assigning an IP address, please see form #1704, the *PAC Manager User's Guide*.

Before you begin, please note the following:

- All SNAP Ethernet-based controllers and brains must be assigned a unique, static IP address. If the network you're using has a Dynamic Host Configuration Protocol (DHCP) server, either assign a static IP address before connecting the device to the network (preferred), or disable the server. (These servers may respond to BootP requests and assign a dynamic address.)

***CAUTION**: To make sure the controller or I/O unit is not on a network with a DHCP server, we recommend you use a crossover cable with a direct connection to assign IP addresses.*

- If you are adding an I/O segment to an existing Ethernet network, your network administrator must provide static IP addresses and subnet masks for the I/O units. If you are creating an independent, dedicated Ethernet network just for I/O, you can choose your own addresses.
- You will need to know the device's MAC address. The MAC address is printed on a label on the side of the device.
- BootP broadcasts cannot get through a firewall in the PC where PAC Manager is running. Make sure any firewall in the computer (such as the built-in firewall in Windows XP) is disabled before you try to assign IP addresses. Firewalls in a router should not be a problem.

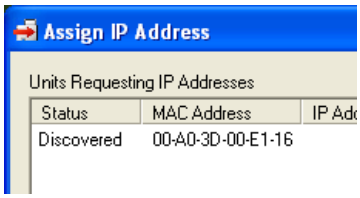
To assign an IP address:

1. Connect the device (controller or brain) using ETHERNET 1. (Only Ethernet 1 sends a BootP request.)

   NOTE: As stated in the CAUTION above, we recommend you use a crossover cable with a direct connection to assign IP addresses.

2. Choose Start→Programs→Opto 22→PAC Project 8.0→PAC Manager.
3. Choose Tools→Assign IP Address.

   In the dialog that opens you should see the device's MAC address. Addresses of other Opto 22 Ethernet-based devices without IP addresses might appear as well.



4. Turn on the controller.
5. Double-click the MAC address of the device to open the Mapping dialog box.

   ***CAUTION:** PAC Manager lists ALL Opto 22 devices sending BootP or DHCP broadcasts. Assign IP addresses only to the ones you know are yours!*

6. Enter an IP Address and Subnet Mask compatible with your network.

   WARNING! *Each device on your network, including computers, routers, controllers, brains, and so on, must have a unique IP address. Failure to assign unique IP addresses may cause catastrophic network or hardware failures. If you don't know which IP addresses are safe to use, check with your system administrator.*

7. When the IP address, subnet mask, and other fields are correct, click OK.

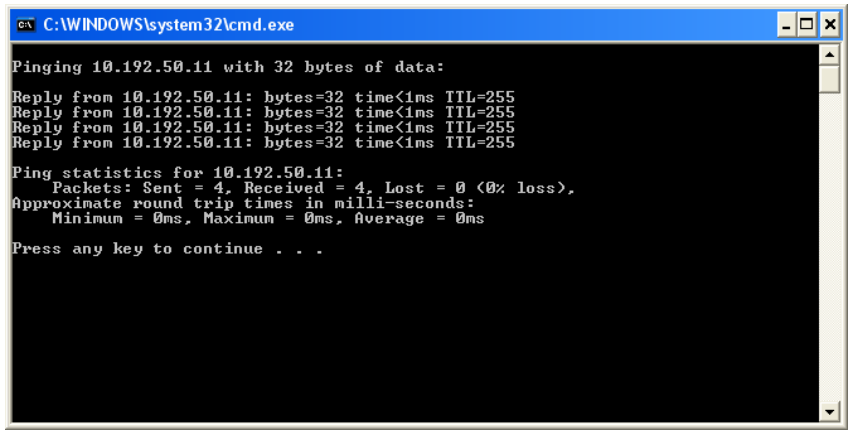
   The new IP address information appears in the upper list in the dialog box, and the device's status changes to Mapped. The address information also appears in the lower list to show that this device has been mapped to this address.

8. With the device still highlighted, click Assign.

   The address is saved to flash memory, and the status changes to Static IP.

9. To verify that the IP address has been successfully assigned, highlight the device in the upper list and click Test.

   A DOS window opens and the IP address is automatically contacted using the PING program. You should see a reply similar to the following:


C:\WINDOWS\system32\cmd.exe
Pinging 10.192.50.11 with 32 bytes of data:
Reply from 10.192.50.11: bytes=32 time<1ms TTL=255
Reply from 10.192.50.11: bytes=32 time<1ms TTL=255
Reply from 10.192.50.11: bytes=32 time<1ms TTL=255
Reply from 10.192.50.11: bytes=32 time<1ms TTL=255
Ping statistics for 10.192.50.11:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
Press any key to continue . . .

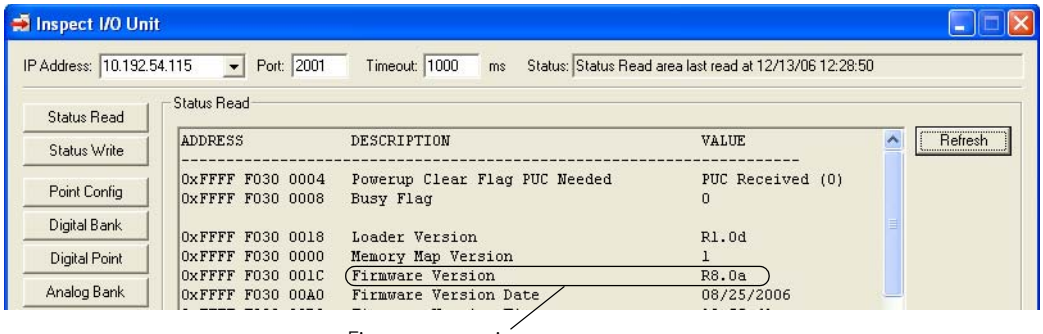
   If you don't see a reply, make sure the subnet mask you've assigned matches the subnet mask on your PC.

10. For future reference, write the IP address next to the MAC address on the white sticker provided on the device, then close the DOS window.
11. Click Close to exit.

Next you will check to make sure the controller or brain is running firmware version 8.0.

Checking the Firmware Version of the Controller or Brain

1. Choose Start→Programs→Opto 22→PAC Project 8.0→PAC Manager.
2. In the PAC Manager main window, click Inspect .
3. In the IP Address field, type the IP address of the SNAP-PAC-R controller or brain, and then click Status Read.



Firmware version

If you don't have firmware version 8.0, new firmware can be downloaded from our website, www.opto22.com. For information on how to load new firmware, see Chapter 6 of the *PAC Manager User's Guide*, form #1704.

Using an Example Strategy

Now you are ready to open one of the following example strategies using PAC Control. The example strategies are available on our website, www.opto22.com.

- **Electronic Gearing Example**: causes axis 0 and axis 1 to rotate for 10 seconds and then stop for 10 seconds.
- **S-Curve Contouring Example**: turns the motor a quarter turn every 10 seconds. This example uses the S-curve profiling mode.
- **Trapezoidal Contouring Example**: Turns the motor a quarter turn every 10 seconds. This example uses the Trapezoidal profiling mode.
- **Velocity Contouring Example**: Turns axis 0 on for 10 seconds then off for 10 seconds.

You can use the example strategies as a starting point for creating your own contol strategies. In particular, take a look at the OptoScript blocks, Axis Setup and Axis 0 Move for each example. The Axis Setup block initializes the SNAP-SCM-BB4 and the Axis 0 Move block makes axis 0 move.

If you don't have PAC Control you can download it from our website, www.opto22.com. For the Electronic Gearing Example you need a stepper motor attached to axis 0 and axis 1. For all other example strategies you only need a stepper motor on axis 0.

To use an example strategy:

1. Unzip the examples from the CD provided to a directory on your local hard drive, and then open one of the strategies.
2. With an example strategy open in Configure mode or Online mode, double-click the Control Engines folder on the Strategy Tree and add the control engine as described in Chapter 5 of the *PAC Control User's Guide*, form #1700.

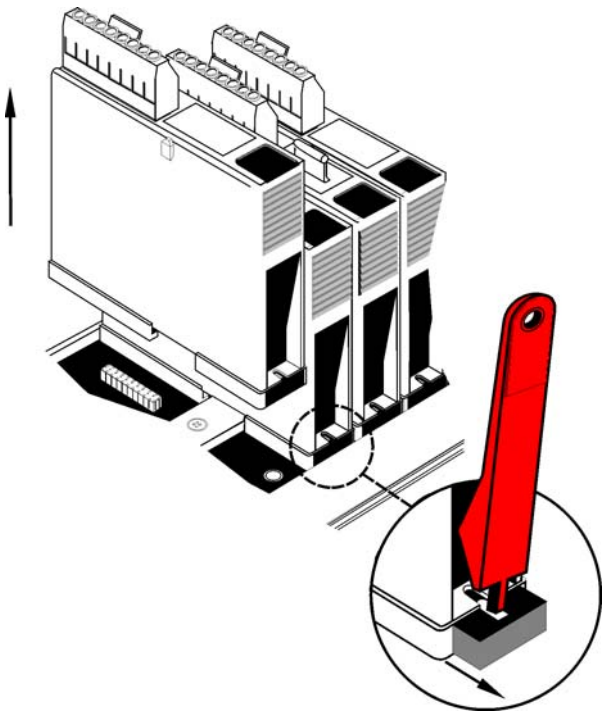
   In the Control Engine dialog box, you only need to enter a descriptive Control Engine Name and the Primary IP address, which is the address of the controller. In the Configure Control Engines dialog box, make sure to select the control engine and click Set Active.
3. Close the dialog boxes for adding a control engine.
4. Choose Mode→Debug.

   The strategy is downloaded to the controller.
5. Click Run Strategy ▶ , and then attend to any dialog boxes that might appear.

   At this point your stepper motor should be spinning.

Removing a Module

1. If the module is held in place with screws, remove them.
2. Holding the SNAP module tool (provided) as shown in the illustration at right, insert it into the notch at the base of the module.
3. Squeeze the module tool against the module to open the release latch, and pull straight up on the module to remove it.



Specifications

This chapter provides specifications of the SNAP-SCM-MCH16 module and SNAP-SCM-BB4 breakout board.

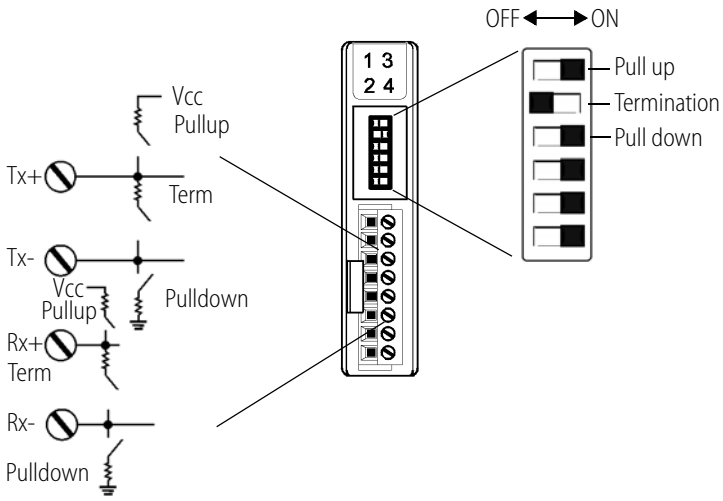
In This Chapter

Module Specifications 20
Breakout Board Specifications 21
Component Connection Schematic 25
Connecting Multiple Breakout Boards 26
Breakout Board Connector Pins 22
Breakout Board Switches 23
RS-422/485 Serial Cable 24
Calculating Power Requirements 27

Module Specifications

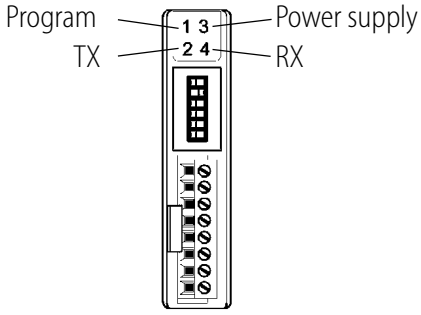
Baud rates	115,200
Parity	Even
Data bits	8 only
Logic supply voltage	5.0 to 5.2 VDC
Logic supply current	250 - 1000 mA DC
Number of ports per module	1
Maximum number of modules per rack	8
Maximum cable length, multi-drop	1,000 feet at 115,200 Baud
I/O processor (brain or on-the-rack controller) compatibility	SNAP-PAC-R1, SNAP-PAC-R2, SNAP-PAC-EB1, or SNAP-PAC-EB2
Operating temperature	0 to 70 °C
Storage temperature	-30 to 85 °C

Module Bias and Termination



Module LEDs

Transmit and receive LEDs are provided as shown in the diagram at right.



LED	Indicates
1	Power Supply Fault
2	Tx
3	Status
4	Rx

Power Supply Fault indicates a fault on the internal power supply of the module (too much draw). This will happen if more than one breakout board is connected without additional power supplies, or there is a short in the system.

Tx: transmitting data

Status: shows module status. This LED blinks twice when the kernel is started.

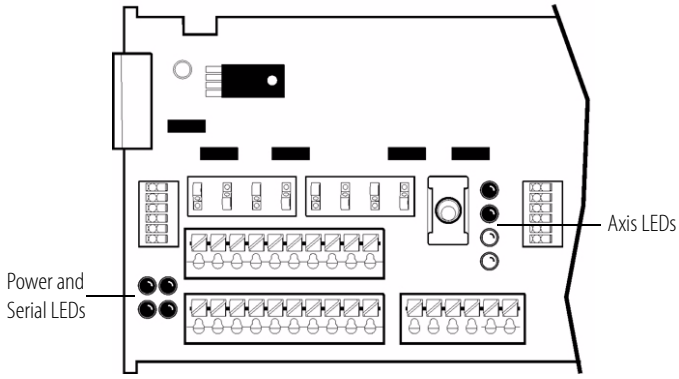
Rx: receiving data

Breakout Board Specifications

Power Requirements	8-24 VDC @ 250mA
Operating Temperature	0 to 70 °C
Relative Humidity	95%, non-condensing

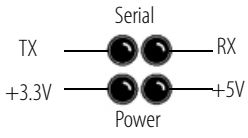
Breakout Board LEDs

There are LED indicators for power and serial, and there are four sets of LEDs for the axis connections, one for each axis.



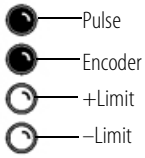
Power and Serial LEDs

The power and serial LEDs indicate the following:

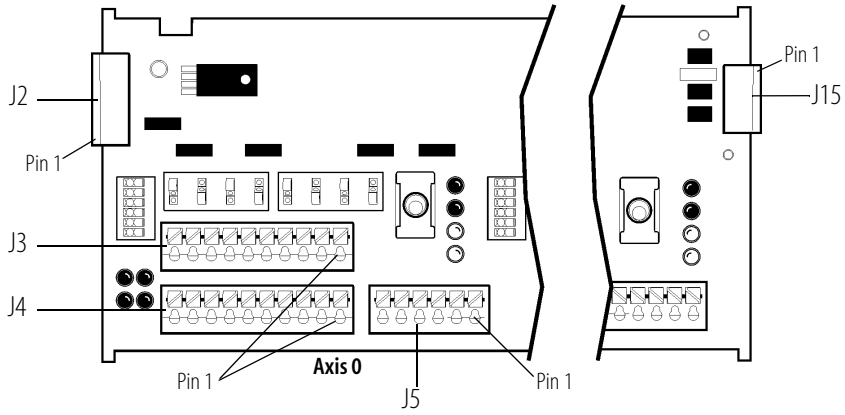


Axis LEDs

There is one set of LED indicators for each axis that indicates the following:



Breakout Board Connector Pins



J15: Auxiliary Power Input

Pin	Description
1	Aux +5Vin
2	Aux +8-24vin
3	GND
4	Chassis GND

J2: Serial Connector

Pin	Description
1	ToHost+
2	ToHost-
3	GND
4	FromHost+
5	FromHost-
6	Chassis GND
7	VMod
8	VMod
9	GND
10	GND

J4 (and J7, J10, & J13): Stepper Motor Outputs

Pin	Description
1	Pulse+
2	Pulse-
3	GND
4	Direction+
5	Direction-
6	AtRest+
7	AtRest-
8	GND
9	AxisOut+
10	AxisOut-

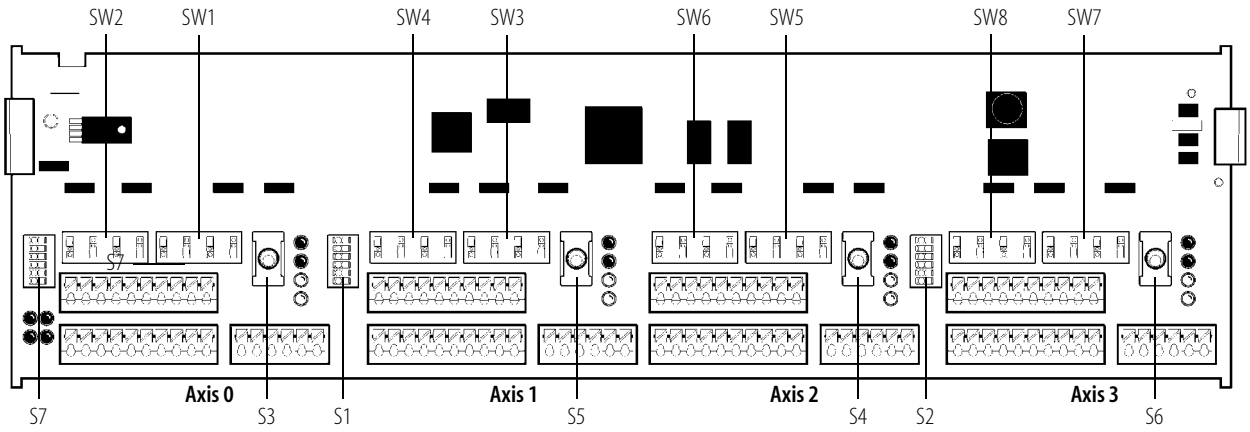
J3 (and J6, J9, & J12): Encoder Signal Inputs

Pin	Description
1	QuadA+
2	QuadA-
3	GND
4	QuadB+
5	QuadB-
6	Index+
7	Index-
8	GND
9	Home+
10	Home-

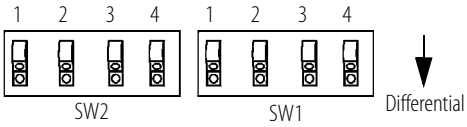
J5 (and J8, J11, & J14): Stepper Motor Inputs

Pin	Description
1	PosLimit
2	GND
3	NegLimit
4	GND
5	AxisIn
6	GND

Breakout Board Switches



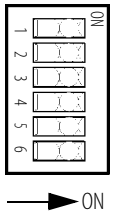
SW1–SW8: Signal Selection for Encoder Inputs



All up=Non-differential
All down=Differential

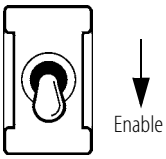
Position	Description
SW1 (and SW3, SW5, & SW7)	
1 & 2	QuadA
3 & 4	QuadB
SW2 (and SW4, SW6, & SW8)	
1 & 2	Index
3 & 4	Home

S1 & S2: Pullup Resistors



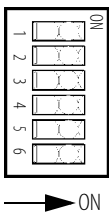
Switch	Axis	Description
S1: J5 & J8 470 Ohm Pull Up Resistor		
1	0	PosLimit
2	0	NegLimit
3	0	AxisIn
4	1	PosLimit
5	1	NegLimit
6	1	AxisIn
S2: J11 & J14 Pull Ups		
1	2	PosLimit
2	2	NegLimit
3	2	AxisIn
4	3	PosLimit
5	3	NegLimit
6	3	AxisIn

S3 (and S5, S4, & S6): Enable/Disable Axis



Position	Enable/Disable
Up	Disable
Middle	Enable
Down	Enable

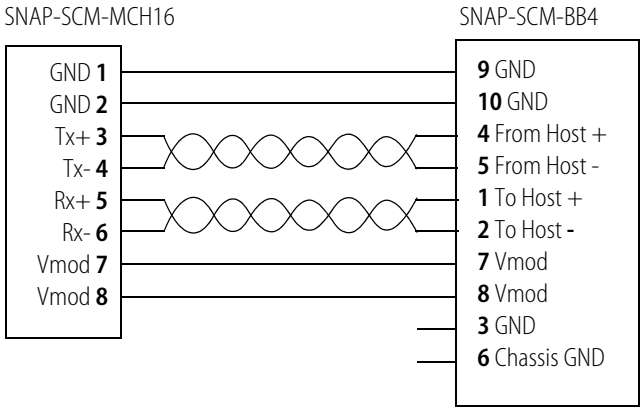
S7: Bias & Termination, Voltage Select, Breakout Board Address



Switch	Description
1	ToHost Termination
2	FromHost Termination
3	VMod Select
4	VMod Select
5	ADDR0
6	ADDR1

RS-422/485 Serial Cable

Use the following schematic to build the RS-422/485 cable that connects the module to the breakout board. The connectors (Opto part numbers 6346 and R80088) are provided in the motion control kit.

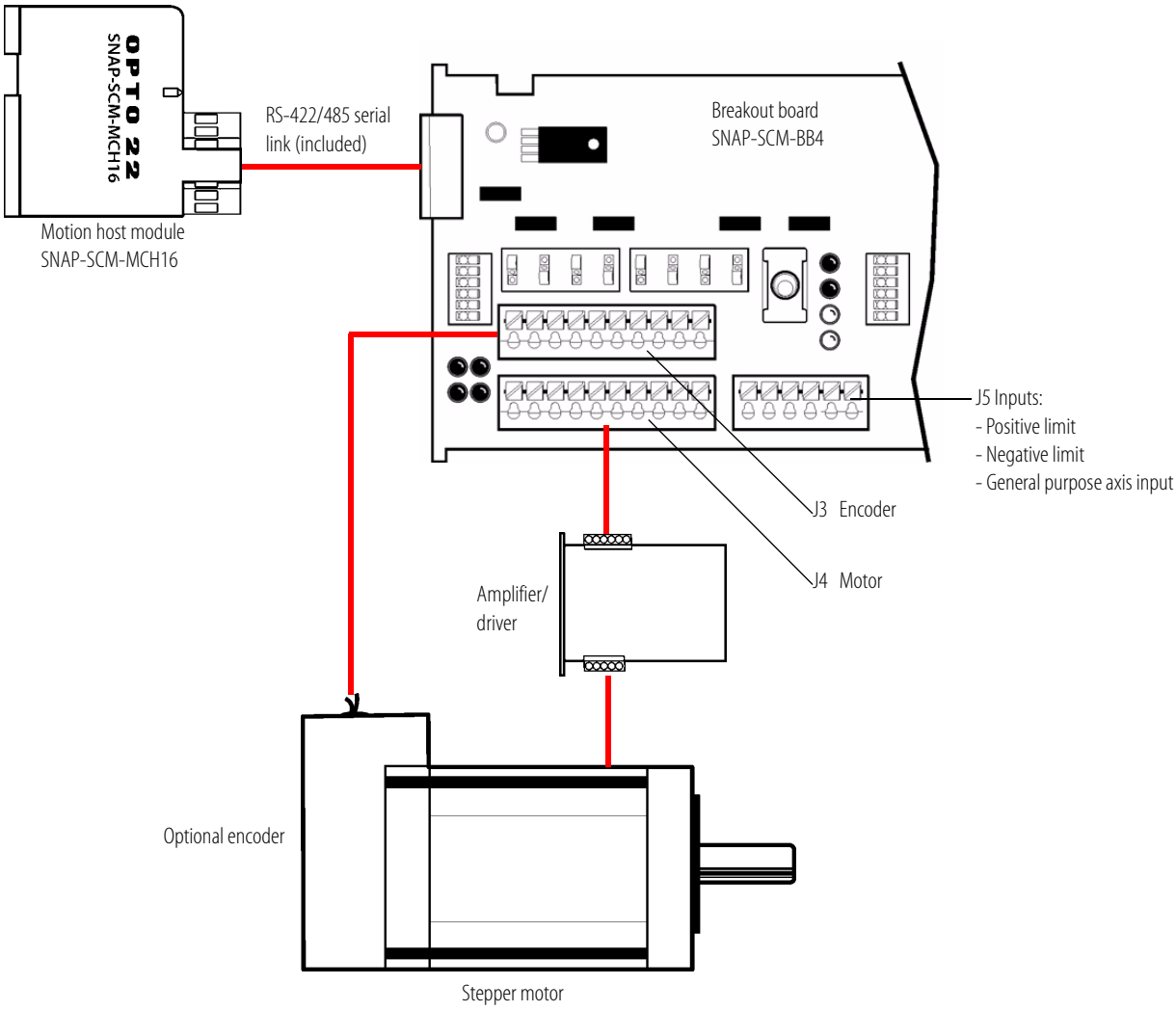


SNAP-SCM-MCH16		SNAP-SCM-BB4	
Description	**Pin**	**Pin**	**Description**
GND	1, 2	3, 9, 10	GND
TX+	3	4	FROM HOST+
TX-	4	5	FROM HOST-
RX+	5	1	TO HOST+
RX-	6	2	TO HOST-
VMOD*	7, 8	7, 8	VMOD*

* VMOD can only power one SNAP-SCM-BB4

Component Connection Schematic

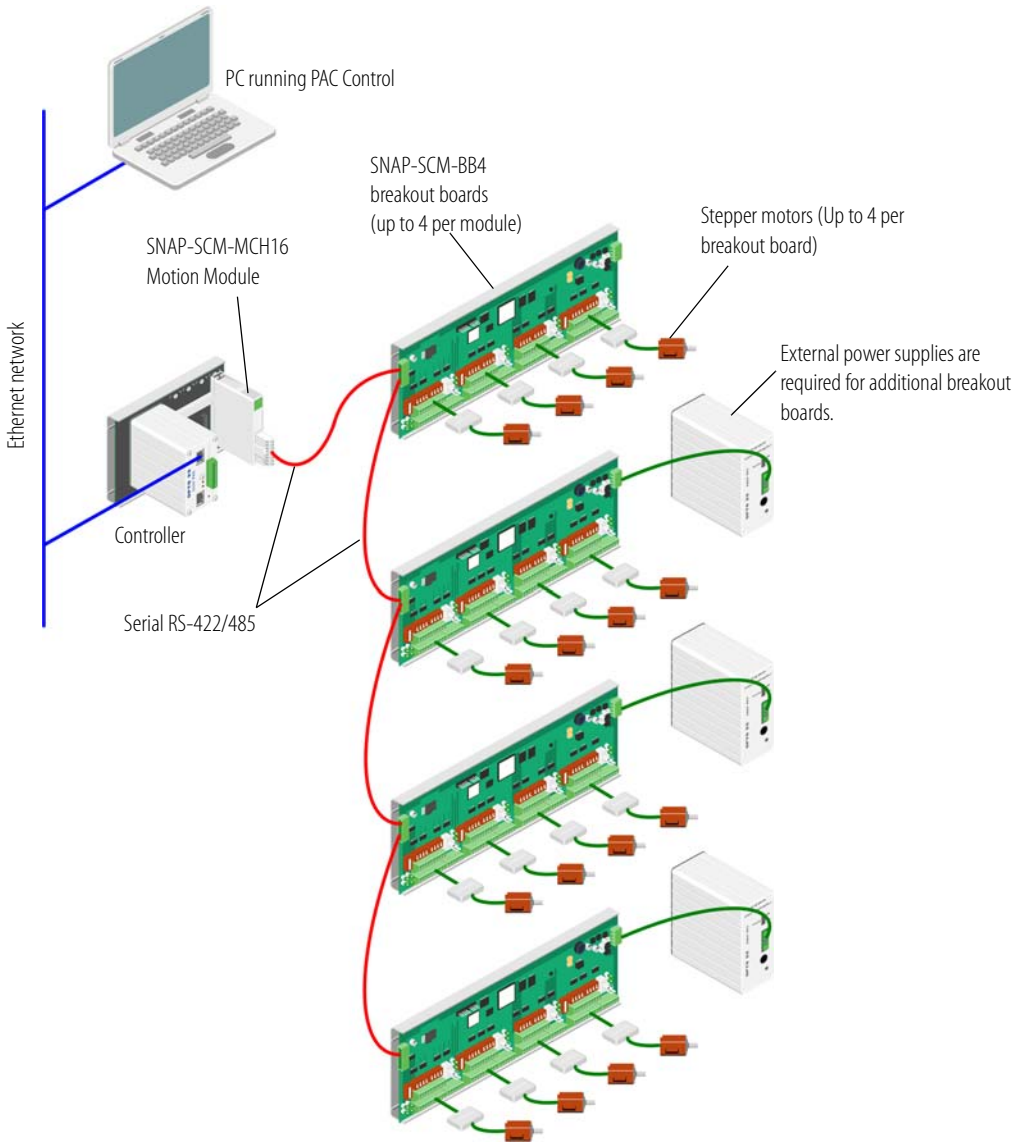
The following schematic shows a basic connection of one module to one breakout board, and includes an optional encoder.



Connecting Multiple Breakout Boards

You can daisy-chain up to four breakout boards as shown here. The first breakout board in the chain receives power from the motion module, while each additional breakout board requires its own external power supply. The boards are connected with serial RS-422/485 cables. You can connect up to four stepper motors per breakout board.

A PC is used to develop and maintain a PAC Control strategy. The strategy is downloaded to the controller where it runs independently of the PC.



Calculating Power Requirements

When you assemble a SNAP rack that includes a SNAP-SCM-MCH16, you need to calculate the power requirements to make sure that the rack's power supply is adequate for the combined current needed by the brain or controller and all the I/O modules. For more information and power requirements worksheets, see the *SNAP I/O Wiring Guide* (form #1403) as well as the wiring appendices in the brain or controller's user's guide.

Using Motion Control Commands

This chapter describes the OptoMotion library of motion commands, which you can use within PAC Control strategies. These commands support many of the Magellan™ Motion Processor commands, giving you the ability to define and acquire motion process data such as position, velocity, acceleration, breakpoints, interrupts, and time intervals. In addition, you can execute motion-related actions such as smooth stops, stepping, and position adjustments.

Commands are sent to the SNAP-SCM-MCH16 motion module via PAC Control. First a TCP Communication Handle is opened to the module. Then you use the Transmit/Receive String to transmit the commands to the module. The motion module supports native English commands used by the Magellan™ Motion Processor. See "Using the Commands" on page 29.

For information on errors returned with checks associated with host I/O commands, "Host I/O Errors" on page 31.

For additional information on using the Magellan™ Motion Processor, see *version 1.x* of the *Magellan™ Motion Processor User's Guide* available on the web at http://www.pmdcorp.com

In This Chapter

Using the Commands.....29
Host I/O Errors.....31

Using the Commands

Commands are sent to the SNAP-SCM-MCH16 motion module via PAC Control. The SNAP-SCM-MCH16 motion module acts as a communication link between Opto 22 controllers and the SNAP-SCM-BB4 breakout board. Using a PAC Control strategy, you open a TCP Communication to the module. Then you use Transmit/Receive to send to the module native English commands used by the ™ Motion Processor. The module converts the text strings to binary commands, which it then passes to the motion processor on the breakout board to control the motion of a stepper motor.

You can also use standard PAC Control to enter the motion control commands. However, this typically takes several blocks in a strategy, so it is usually better to use OptoScript. For detailed descriptions of the available commands, see the commands starting on page 37.

Entering Commands in OptoScript

A motion control command is entered as a text string in OptoScript using the TransmitReceiveString command, which is the same as Transmit/Receive String in standard PAC Control. This command sends the motion control command to the processor and then waits for a response. For more information on the Transmit/Receive String command, see the *PAC Control Command Reference*, form #1701. For information on motion control strategy examples available on the Opto 22 website, see "Using an Example Strategy" on page 16.

To enter a motion control command, use the > symbol at the beginning of the command and enclose the command in double quotes, as shown in the following example:

Syntax:

`TransmitReceiveString("><i>Command</i>", <i>Communication Handle, Put Result in</i>)`

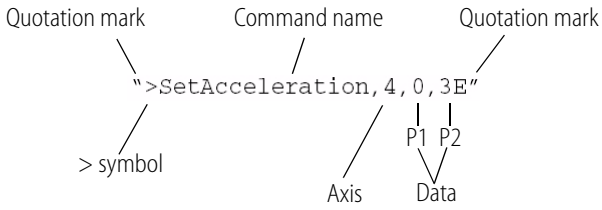
PAC Control command | Motion control command | Communication Handle | Put Result in

```
Status = TransmitReceiveString(">SetAcceleration,4,0,3E", ComHandleVariable, RespStringVariable
```

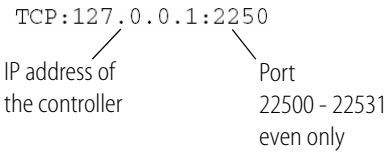
For the PAC Control command use TransmitReceiveString. Note that the Put Result in parameter is the response from the motion control command. The following examples show the SetAcceleration and Get Acceleration commands.

Example 1: SetAcceleration.

Motion control command:



Communication Handle:



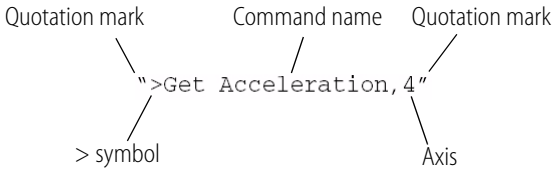
For more information on communication handles, see Chapter 10 in the *PAC Control User's Guide*.

Put Result in:

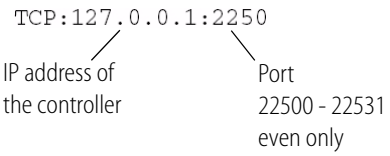
`Status` If the status byte returns a 00, then there is no error. Any other response indicates an error.

Example 2: Get Acceleration.

Motion control command:

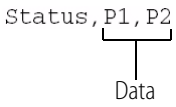


Communication Handle:



For more information on communication handles, see Chapter 10 in the *PAC Control User's Guide*.

Put Result in:



If the status byte returns a 00, then there is no error. Any other response indicates an error.

Host I/O Errors

There are a number of checks made by the motion processor on the commands sent to it. These checks improve safety of the motion system by eliminating incorrect command data values. All such checks associated with host I/O commands are referred to as host I/O errors. If any such error occurs, bit 13 of the I/O status read word (see previous section for definition of this word) is set. To determine the error's cause, the command GetHostIOError is used. Executing the GetHostIOError command also clears both the error code and the I/O error bit in the I/O status read word.

I/O error codes which are returned by the GetHostIOError command are detailed in the following table.

Code	Indication	Cause
0	No error	No error condition
1	Magellan reset	Default value of error code on reset or power-up.
2	Invalid instruction	Instruction is not valid in the current context, or an illegal instruction code has been detected.
3	Invalid axis	The axis number contained in the upper bits of the instruction word is not supported by this motion processor.

Code	Indication	Cause
4	Invalid parameter	The parameter value sent to the motion processor was out of its acceptable range.
5	Trace running	An instruction was issued that would change the state of the tracing mechanism while the trace is running. Instructions which can return this error are SetTraceVariable, SetTrace-Mode & SetTracePeriod.
6	Reserved	--
7	Block bound exceeded	1. The value sent by SetBufferLength or SetBufferStart would create a memory block which extends beyond the allowed limits of 400h - 7FFFFFFFh. 2. Either SetBufferReadIndex or SetBufferWriteIndex sent an index value greater than or equal to the block length.
8	Trace zero	SetTraceStart Immediate was issued, but the length of the trace buffer is currently set to zero.
9	Bad checksum *(Serial port only)*	The checksum complied and returned by Magellan does not match the checksum which was sent by the host.
Ah	Not primary port	A prohibited instruction (one which can be executed only through the primary port) was issued through the diagnostic port.
Bh	Negative velocity	An attempt was made to set a negative velocity without the axis being in velocity contouring profile mode.
Ch	S-curve change	The axis is currently executing an S-curve profile move and an attempt was made to change the profile parameters. This is not permitted.
Dh	Limit event pending	A limit switch event has occurred.
Eh	Move into limit	An attempt was made to execute a move without first clear-ing the limit bit(s) in the event status register.

Command Reference

This command reference provides a list of the motion control commands by group, and a detailed explanation of each command listed in alphabetical order.

In This Chapter

Commands Listed by Group.....33
Command Details 35
Commands in Alphabetical Order 37

Commands Listed by Group

The following Magellan™ Motion Processor commands are supported in PAC Control using OptoScript.

Commands	Description
Breakpoints and Interrupts	
ClearInterrupt	Reset interrupt line.
Set/GetBreakPoint	Set/Get breakpoint type.
Set/GetBreakPointValue	Set/Get breakpoint comparison value.
GetInterruptAxis	Get the axes with pending interrupts.
Set/GetInterruptMask	Set/Get interrupt mask.
Digital Servo Filter	
ClearPositionError	Set position error to 0.
Set/GetAutoStopMode	Set/Get auto stop on position error (on or off).
GetPositionError	Get the position error.
Set/GetPositionErrorLimit	Set/Get the maximum position error limit.
Encoder	
AdjustActualPosition	Sum the specified offset with the actual encoder position.
Set/GetActualPosition	Set/Get the actual encoder position.
Set/GetActualPostionUnits	Set/Get the unit type returned for the actual encoder position.
GetActualVelocity	Get the actual encoder velocity.

Commands	Description
Set/GetCaptureSource	Set/Get the capture source (home or index).
GetCaptureValue	Get the position capture value, and reset the capture.
Set/GetEncoderModulus	Set/Get the full scale range of the parallel-word encoder
Set/GetEncoderSource	Set/Get the encoder type.
Set/GetEncoderToStepRatio	Set/Get encoder count to step ratio.
External RAM	
Set/GetBufferLength	Set/Get the length of a memory buffer.
Set/GetBufferReadIndex	Set/Get the buffer read pointer for a particular buffer.
Set/GetBufferStart	Set/Get the start location of a memory buffer.
Set/GetBufferWriteIndex	Set/Get the buffer write pointer for a particular buffer.
ReadBuffer	Read a long word value from a buffer memory locations.
WriteBuffer	Write a long word value to a buffer memory location.
Motor Output	
Set/GetMotorMode	Set/Get motor loop mode.
Set/GetMotorType	Set/Get motor type for axis.
Set/GetStepRange	Set/Get the allowable range (in kHz) for step output generation.
Profile Generation	
Set/GetAcceleration	Set/Get acceleration limit.
GetCommandedAcceleration	Get commanded (instantaneous desired) acceleration
GetCommandedPosition	Get commanded (instantaneous desired) position.
GetCommandedVelocity	Get commanded (instantaneous desired) velocity.
Set/GetDeceleration	Set/Get deceleration limit.
Set/GetGearMaster	Set/Get the electronic gear mode master axis and source (actual or target-based).
Set/GetGearRatio	Set/Get commanded electronic gear ratio.
Set/GetJerk	Set/Get jerk limit.
Set/GetPosition	Set/Get the destination position.
Set/GetProfileMode	Set/Get the profile mode (S-curve, trapezoidal, velocity-contouring, or electronic gear).
Set/GetStartVelocity	Set/Get start velocity.
Set/GetStopMode	Set/Get stop command; abrupt, smooth, or none.
Set/GetVelocity	Set/Get velocity limit.
MultiUpdate	Forces buffered command values to become active for multiple axes.
Update	Forces buffered command values to become active.
ServoLoopControl	
Set/GetAxisMode	Set/Get the axis operation mode (enabled or disabled).
Set/GetLimitSwitchMode	Set/Get the limit switch mode (on or off).
Set/GetMotionCompleteMode	Set/Get the motion complete mode (target-based or actual).
Set/GetSampleTime	Set/Get servo loop sample time.
Set/GetSettleTime	Set/Get the axis-settled time.
Set/GetSettleWindow	Set/Get the settle-window boundary value.
GetTime	Get current chip set time (number of servo loops).
Set/GetTrackingWindow	Set/Get the tracking window boundary value.

Commands	Description
Status Registers and AxisOut Indicator	
GetActivityStatus	Get activity status register.
Set/GetAxisOutSource	Set/Get axis out signal monitor source.
GetEventStatus	Get event status register.
GetSignalStatus	Get the signal status register.
ResetEventStatus	Reset bits in event status register.
Set/GetSignalSense	Set/Get the interpretation of the signal status bits.

Command Details

For each command in the following section, the instruction mnemonic is shown in the command heading at the left. Certain parameters and other data written to the motion processor are buffered. That is, they are not acted upon until the next Update or MultiUpdate command is executed. These parameters are identified by the word **buffered** in the command heading at the right.

The details for each command are as follows:

Syntax

The instruction mnemonic and its required arguments are shown with all arguments separated by commas.

Arguments

Encoded-field arguments are packed into a single 16-bit data word. The **Name** of the argument is that shown in the generic syntax.

Instance is the mnemonic used to represent the data value. **Encoding** is the value assigned to the field for that instance.

For numeric arguments, the parameter **Value**, the **Type** (signed or unsigned integer) and **Range** of acceptable values are given. Numeric arguments may require one or two data words. For 32-bit arguments, the high-order part is transmitted first.

You must specify the location of the stepper for each command. The number of the stepper motor identifies one of four SNAP-SCM-BB4 breakout boards as follows:

Stepper Motor Location	Breakout Board Address
0 to 3	0
4 to 7	1
8 to B	2
C to F	3

Data Structure

This is a graphic representation of the 16-bit words transmitted in the packet: the instruction, which is identified by its name, followed by 1, 2, or 3 data words. Bit numbers are shown directly below each word. For each field in a word, only the high and low bits are shown. For 32-bit numeric data, the high-order bits are numbered from 16 to 31, the low-order bits from 0 to 15.

Argument names are shown in their respective words or fields.

For data words, the direction of transfer-read or write-is shown at the left of the word's diagram.

Unused bits are shaded. All unused bits must be 0 in data words and instructions sent (written) to the motion processor.

Description Describes what the instruction does and any special information relating to the instruction.

Restrictions Describes the circumstances in which the instruction is not valid, that is, when it should not be issued. For example, velocity, acceleration, deceleration, and jerk parameters may not be issued while an S-curve profile is being executed.

OptoScript Example The syntax of the text string entered within the TransmitReceiveString OptoScript command, which is sent to the motion module for conversion to the corresponding PMD C-Motion command for implementation by the motion control processor.

See Also Refers to related instructions.

Commands in Alphabetical Order

AdjustActualPosition

Syntax

AdjustActualPosition, *axis, P1, P2*

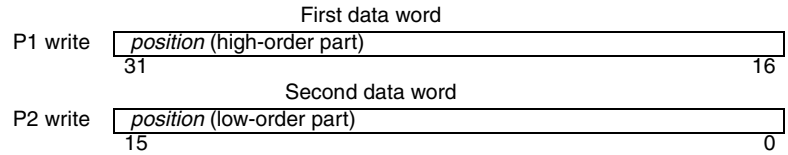
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

	Type	Range	Scaling	Units
position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts/cycle^2 microsteps/cycle

Data Structure

AdjustActualPosition



Description

The position specified as the parameter to AdjustActualPosition is summed with the actual position register (encoder position) for the specified axis. This has the effect of adding or subtracting an offset to the current actual position. At the same time, the commanded position is replaced by the new actual position value minus the position error. This prevents a servo "bump" when the new axis position is established. The destination position (see "SetPosition, GetPosition" on page 95) is also modified by this amount so that no trajectory motion will occur when the update instruction is issued. In effect, this instruction establishes a new reference position from which subsequent positions can be calculated. It is commonly used to set a known reference position after a homing procedure. The position error is also zeroed.

AdjustActualPosition takes effect immediately, it is not buffered.

OptoScript Example

```
Status=TransmitReceiveString
(">AdjustActualPosition,4,0,3E",ComHandle,ResponseString)
```

See Also

GetPositionError (page 51), GetActualVelocity (page 42), SetActualPositionUnits, GetActualPositionUnits (page 63), SetActualPosition, GetActualPosition (page 61)

ClearInterrupt

Syntax

ClearInterrupt, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Description

ClearInterrupt resets the HostInterrupt signal to its inactive state. If interrupts are still pending, the HostInterrupt line will return to its active state within one chip cycle. Refer to Set/GetSampleTime for information on chip cycle timing (page 98). It is used after an interrupt has been recognized and processed by the host. This command does not affect the event status register. The ResetEventStatus command should be issued prior to the ClearInterrupt command to clear the condition which generated the interrupt. The ClearInterrupt command has no effect if it is executed when no interrupts are pending.

The axis number is not used.

OptoScript Example

```
Status=TransmitReceiveString
(">ClearInterrupt,0",ComHandle,ResponseString)
```

See Also

GetInterruptAxis (page 50), SetInterruptMask, GetInterruptMask (page 87), ResetEventStatus (page 58)

ClearPositionError

buffered

Syntax

ClearPositionError, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Description

ClearPositionError sets the profile's commanded position equal to the actual position (encoder input), thereby clearing the position error for the specified axis. This command can be used when the axis is at rest, or when it is moving.

Restrictions

ClearPositionError is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction.

This command should not be sent while the chip is executing a move using the S-curve profile mode.

OptoScript Example

```
Status=TransmitReceiveString
(">ClearPositionError,1",ComHandle,ResponseString)
```

See Also

GetPositionError (page 51), MultiUpdate (page 55), SetPositionErrorLimit, GetPositionErrorLimit (page 96)

GetActivityStatus

Syntax GetActivityStatus, *axis, P1*

Arguments

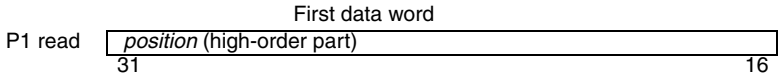
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	
status	unsigned 32 bits	*see below*

Data Structure

GetActivityStatus



Description

GetActivityStatus reads the 16-bit activity status register for the specified axis. Each of the bits in this register continuously indicate the state of the motion processor without any action on the part of the host. There is no direct way to set or clear the state of these bits, since they are controlled by the motion processor.

The following table shows the encoding of the data returned by this command

Name	Bits	Description
reserved	6	Not used; may be 0 or 1.
Phasing initialized	0	Set to 1 if phasing is initialized (Brushless DC axes only).
At maximum velocity	1	Set to 1 when the trajectory is at maximum velocity. This bit is determined by the trajectory generator, not the actual encoder velocity.
Tracking	2	Set to 1 when the axis is within the tracking window.
Current profile mode	3-5	Contains trajectory mode encoded as follows:
		bit 5 / bit 4 / bit 3 / Profile Mode 0 / 0 / 0 / trapezoidal 0 / 0 / 1 / velocity contouring 0 / 1 / 0 / s-curve 0 / 1 / 1 / electronic gear
Axis settled	7	Set to 1 when the axis is settled.
Motor on/off	8	Set to 1 when motor mode is on, 0 when off.
Position capture	9	Set to 1 when a value has been captured by the high speed position capture hardware but has not yet been read.
In-motion	10	Set to 1 when the trajectory generator is executing a profile.
In positive limit	11	Set to 1 when the positive limit switch is active.
In negative limit	12	Set to 1 when the negative limit switch is active.

Name	Bits	Description
Profile segment	13-15	When the profile mode is S-curve, it contains the profile segment number 1-7 while profile is in motion, and contains a value of 0 when the profile is at rest. This field is undefined when using the Trapezoidal and Velocity Contouring profile modes.

OptoScript Example

```
Status=TransmitReceiveString
(">GetActivityStatus,2",ComHandle,ResponseString)
```

See Also GetEventStatus (page 47), GetSignalStatus (page 52)

GetActualVelocity

Syntax

GetActualVelocity, *axis*

Arguments

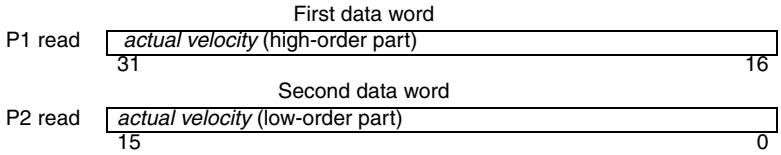
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	Range	Scaling	Units
velocity	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle^2

Data Structure

GetActualVelocity



Description

GetActualVelocity reads the value of the velocity for the specified axis. The actual velocity is derived by subtracting the actual position during the previous chip cycle from the actual position for this chip cycle. The result of this subtraction will always be integer because position is always integer. As a result the value returned by GetActualVelocity will always be a multiple of 65,536 since this represents a value of one in the 16.16 number format. The low word is always zero.

This value is the result of the last encoder input, so it will be accurate to within one cycle.

Scaling example: If a value of 1,703,936 is retrieved by the GetActualVelocity command (high word: 01Ah, low word: 0h) this corresponds to a velocity of 1,703,936/65,536 or 26 counts/cycle. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Restrictions

The actual velocity is derived by subtracting the actual position during the previous chip cycle from the actual position for this chip cycle. The result of this subtraction will always be integer because position is always integer. As a result the value returned by GetActualVelocity will always be a multiple of 65,536 since this represents a value of one in the 16.16 number format. The low word is always zero.

OptoScript Example

```
Status=TransmitReceiveString
(">GetActualVelocity,3",ComHandle,ResponseString)
```

See Also

GetCommandedVelocity (page 46)

GetCaptureValue

Syntax

GetCaptureValue, *axis*

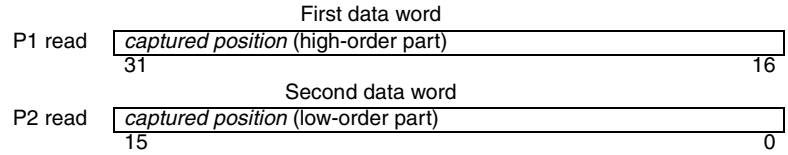
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
captured position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts microsteps

Data Structure

GetCaptureValue



Description

GetCaptureValue returns the contents of the position capture register for the specified axis. This command also resets bit 9 of the activity status register; thus allowing another capture to occur.

If actual position units is set to steps, the returned position will be in units of steps.

OptoScript Example

```
Status=TransmitReceiveString
(">GetCaptureValue,4",ComHandle,ResponseString)
```

See Also

SetCaptureSource, GetCaptureSource (page 79), SetActualPositionUnits, GetActualPositionUnits (page 63), GetActivityStatus (page 40)

GetCommandedAcceleration

Syntax

GetCommandedAcceleration, *axis*

Arguments

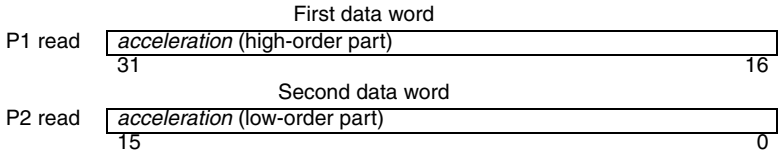
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	Range	Scaling	Units
acceleration	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle^2 microsteps/cycle

Data Structure

GetCommandedAcceleration



Description

GetCommandedAcceleration returns the commanded acceleration value for the specified axis. Commanded acceleration is the instantaneous acceleration value output by the trajectory generator.

Scaling example: If a value of 114,688 is retrieved using this command then this corresponds to 114,688/ 65,536 = 1.750 counts/cycle2 acceleration value. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

OptoScript Example

```
Status=TransmitReceiveString
(">GetCommandedAcceleration,5",ComHandle,ResponseString)
```

See Also

GetCommandedPosition (page 45), GetCommandedVelocity (page 46)

GetCommandedPosition

Syntax

GetCommandedPosition, *axis*

Arguments

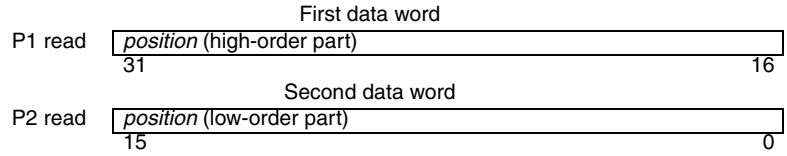
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	Range	Scaling	Units
position	signed 32 bits	-2^{31} to $2^{31}-1$	unity	counts microsteps

Data Structure

GetCommandedPosition



Description

GetCommandedPosition returns the commanded position for the specified axis. Commanded position is the instantaneous position value output by the trajectory generator.

This command functions in all profile modes.

OptoScript Example

```
Status=TransmitReceiveString
(">GetCommandedPosition,6",ComHandle,ResponseString)
```

See Also

GetCommandedAcceleration (page 44), GetCommandedVelocity (page 46)

GetCommandedVelocity

Syntax

GetCommandedVelocity, *axis*

Arguments

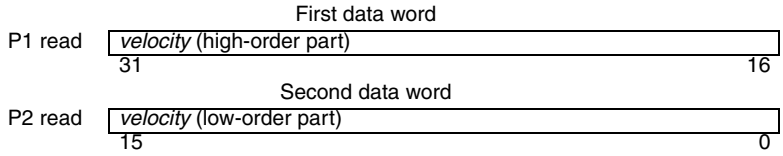
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	Range	Scaling	Units
velocity	signed 32 bits	-2^{31} to $2^{31}-1$	$1/2^{16}$	counts/cycle microsteps/cycle

Data Structure

GetCommandedVelocity



Description

GetCommandedVelocity returns the commanded velocity value for the specified axis. Commanded velocity is the instantaneous velocity value output by the trajectory generator.

Scaling example: If a value of -1,234,567 is retrieved using this command (FFEDh in high word, 2979h in low word) then this corresponds to -1,234,567/65,536 = -18.8380 counts/cycle velocity value. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

OptoScript Example

```
Status=TransmitReceiveString
(">GetCommandedVelocity,7",ComHandle,ResponseString)
```

See Also

GetCommandedAcceleration (page 44), GetCommandedPosition (page 45)

GetEventStatus

Syntax

GetEventStatus, *axis*

Arguments

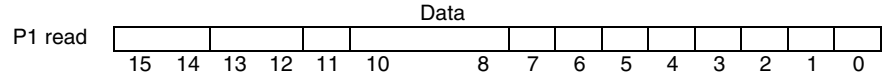
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	
status	unsigned 16 bits	*see below*

Data Structure

GetEventStatus



Description

GetEventStatus reads the event register for the specified axis. All of the bits in this status word are set by the motion processor and cleared by the host. To clear these bits, use the ResetEventStatus command. The following table shows the encoding of the data returned by this command.

Name	Bit(s)	Description
Motion complete	0	Set to 1 when motion has completed. SetMotionCompleteMode determines if this bit is based on the trajectory generator position or the encoder position.
Wrap-around	1	Set to 1 when the actual (encoder) position has wrapped from maximum allowed position to minimum, or vice versa.
Breakpoint 1	2	Set to 1 when breakpoint 1 has been triggered.
Capture received	3	Set to 1 when a position capture has occurred.
Motion error	4	Set to 1 when a motion error has occurred.
In positive limit	5	Set to 1 when the axis has entered a positive limit switch.
In negative limit	6	Set to 1 when the axis has entered a negative limit switch.
Instruction error	7	Set to 1 when an instruction error has occurred.
reserved 8 - 10	8 - 10	Not used; may be 0 or 1.
Commutation error	11	Set to 1 when a commutation error has occurred.
reserved 12 - 13	12 - 13	Not used; may be 0 or 1.
Breakpoint 2	14	Set to 1 when breakpoint 2 has been triggered.
reserved 15	15	Not used; may be 0 or 1.

OptoScript Example

```
Status=TransmitReceiveString
(">GetEventStatus,8",ComHandle,ResponseString)
```

See Also

GetActivityStatus (page 40), GetSignalStatus (page 52)

GetHostIOError

Syntax

GetHostIOError, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

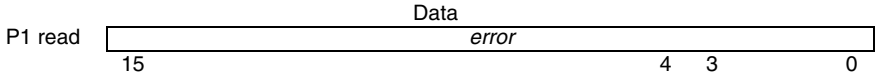
Name	Instance	Encoding
axis mask	None	0
	Axis0mask	1
	Axis1mask	2
	Axis2mask	4
	Axis3mask	8

Name Type Range

error unsigned 16 bits 0 - Eh

Packet Structure

GetHostIOError



Description

GetHostIOError returns the code for the last host I/O error, and then resets the error to zero. Generally, this command is issued only after the instruction error bit in the event status register indicates there was an I/O error. It also resets the HostIOError bit in the I/O status read word to zero.

The error codes are encoded as defined below:

Error Code	Encoding
No error	0
Processor Reset	1
Invalid instruction	2
Invalid axis	3
Invalid parameter	4
Trace running	5
reserved	6
Block out of bounds	7
Trace buffer zero	8
Bad serial checksum	9
reserved	Ah
Invalid negative value	Bh
Invalid parameter change	Ch

Error Code	Encoding
Invalid move after limit condition	Dh
Invalid move into limit	Eh

OptoScript Example

```
Status=TransmitReceiveString
(">GetIOHostError,0",ComHandle,ResponseString)
```

See also GetEventStatus (page 47)

GetInterruptAxis

Syntax

GetInterruptAxis, *axis*

Arguments

none

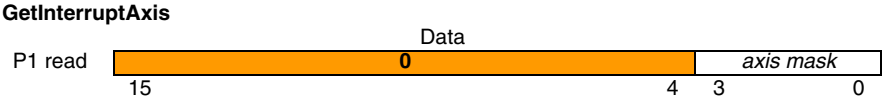
Returned Data

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
axis mask	None	0
	Axis0mask	1
	Axis1mask	2
	Axis2mask	4
	Axis3mask	8

Data Structure

GetInterruptAxis



Description

GetInterruptAxis returns a field which identifies all axes on the breakout board with pending interrupts. Axis numbers are assigned to the low-order four bits of the returned word; bits corresponding to interrupting axes are set to 1. If there are no pending interrupts, the returned word is 0. If any axis has a pending interrupt, the *HostInterrupt* signal will be in an active state.

OptoScript Example

```
Status=TransmitReceiveString
(">GetInterruptAxis,9",ComHandle,ResponseString);
```

See Also

ClearInterrupt (page 38), SetInterruptMask, GetInterruptMask (page 87)

GetPositionError

Syntax

GetPositionError, *axis*

Arguments

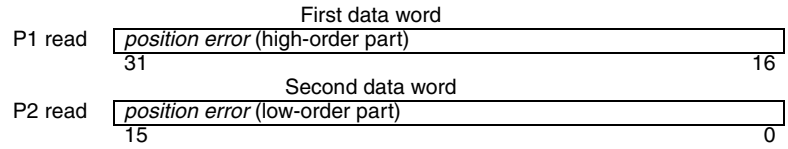
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	Range	Scaling	Units
velocity	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts microsteps

Data Structure

GetPositionError



Description

GetPositionError returns the position error of the specified axis. The error is the difference between the actual position (encoder position) and the commanded position (instantaneous output of the trajectory generator). When used with the motor type set to microstepping, or pulse and direction, the error is defined as the difference between the encoder position (represented in steps), and the commanded position (instantaneous output of the trajectory generator).

OptoScript Example

```
Status=TransmitReceiveString
(">GetPositionError,9",ComHandle,ResponseString)
```

See Also

SetPosition, GetPosition (page 95), SetPositionErrorLimit, GetPositionErrorLimit (page 96)

GetSignalStatus

Syntax

GetSignalStatus, *axis*

Arguments

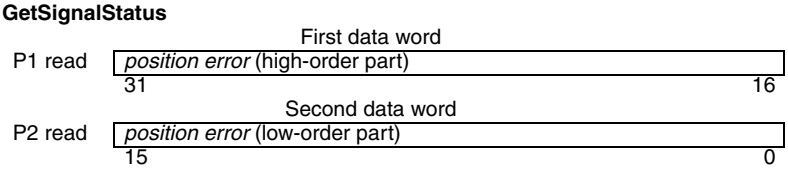
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type
see below	unsigned 16 bits

Data Structure

GetSignalStatus



Description

GetSignalStatus returns the contents of the signal status register for the specified axis. The signal status register contains the value of the various hardware signals connected to each axis of the motion processor. The value read is combined with the signal sense register (see "SetSignalSense, GetSignalSense" on page 102) and then returned to the user. For each bit in the Signal Sense register that is set to 1, the corresponding bit in the GetSignalStatus command will be inverted. Therefore, a low signal will be read as 1, and a high signal will be read as a 0. Conversely, for each bit in the signal sense register that is set to 0, the corresponding bit in the GetSignalStatus command is not inverted. Therefore, a low signal will be read as 0, and a high signal will be read as a 1.

All of the bits in the GetSignalStatus command are inputs, except for AxisOut. The value read for this bit is equal to the value output by the axis out mechanism. See "SetAxisOutSource, GetAxisOutSource" on page 66 for more details. The bit definitions are as follows:

Description	Bit Number
Encoder A	0
Encoder B	1
Encoder Index	2
Encoder Home	3
Positive limit	4
Negative limit	5
AxisIn	6
reserved	7–9
AxisOut	10
reserved	11–15

OptoScript Example

```
Status=TransmitReceiveString
(">GetSignalStatus,A",ComHandle,ResponseString)
```

See Also GetActivityStatus (page 40), GetEventStatus (page 47), SetSignalSense, GetSignalSense (page 102)

GetTime

Syntax

GetTime, *axis*

Arguments

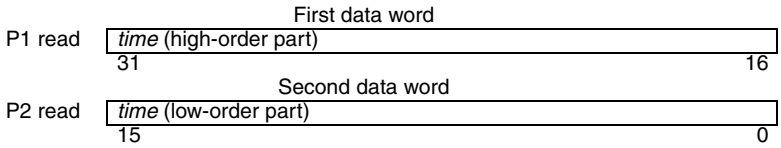
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Returned Data

Name	Type	Range	Scaling	Units
velocity	signed 32 bits	-2^{31} to $2^{31}-1$	unity	counts

Data Structure

GetTime



Description

GetTime returns the number of cycles which have occurred since the motion processor was last reset.

OptoScript Example

```
Status=TransmitReceiveString (">GetTime,B",ComHandle,ResponseString)
```

MultiUpdate

Syntax MultiUpdate, *axis*, *P1*

Arguments

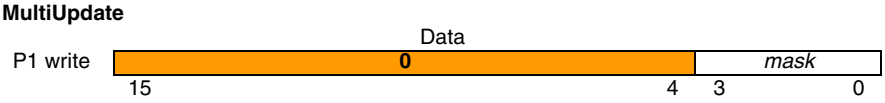
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
mask	None	0
	Axis1mask	1
	Axis2mask	2
	Axis3mask	4
	Axis4mask	8

Returned data none

Data Structure

MultiUpdate



Description MultiUpdate causes an Update to occur on all axes whose corresponding bit is set to 1 in the mask argument. After this command is executed, and for those axes which are selected using the mask, all buffered data parameters are copied into the corresponding run-time registers. The following table shows the buffered commands and variables which are made active as a result of the Update command.

Type	Command
General	ClearPositionError
Trajectory	Acceleration Deceleration GearRatio Jerk Position ProfileMode StopMode Velocity
Servo	DerivativeTime IntegrationLimit Kaff Kd Ki Kp Kvff
Motor	MotorCommand

OptoScript Example

```
Status=TransmitReceiveString
(">MultiUpdate,B,P1",ComHandle,ResponseString)
```

See Also Update (page 111)

ReadBuffer

Syntax

ReadBuffer, *axis*, *P1*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

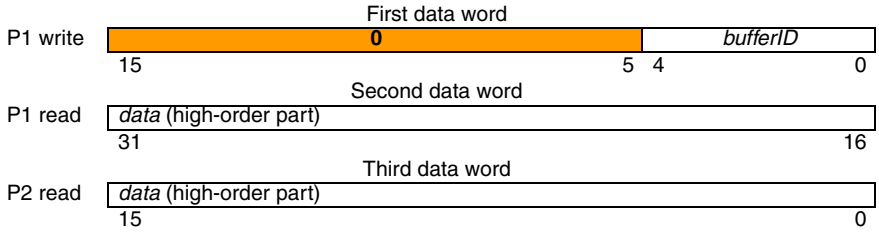
Name	Type	Range
bufferID	unsigned 16 bits	*-0* to 31

Returned Data

Name	Type	Range
data	signed 32 bits	-2^{31} *to* $2^{31}-1$

Data Structure

ReadBuffer



Description

ReadBuffer returns the 32-bit contents of the location pointed to by the read buffer index in the specified buffer. After the contents have been read, the read index is incremented by 1; if the result is equal to the buffer length (set by SetBufferLength), the index is reset to 0. The read index is automatically changed at the completion of a trace when in rolling trace mode.

OptoScript Example

```
Status=TransmitReceiveString
(">ReadBuffer,C,P1",ComHandle,ResponseString)
```

See Also

SetBufferReadIndex, GetBufferReadIndex (page 74), WriteBuffer (page 112), SetBufferStart, GetBufferStart (page 76), SetBufferLength, GetBufferLength (page 72)

ResetEventStatus

Syntax

ResetEventStatus, *axis, P1*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

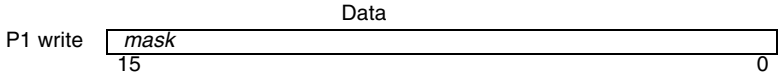
Name	Instance	Encoding
mask	Motion Complete	0001h
	Wrap-Around	0002h
	Breakpoint 1	0004h
	Capture Received	0008h
	Motion Error	0010h
	In positive limit	0020h
	In negative limit	0040h
	Instruction error	0080h
	Commutation error	0800h
	Breakpoint 2	4000h

Returned Data

none

Packet Structure

ResetEventStatus



Description

ResetEventStatus clears (sets to 0), for the specified axis, each bit in the event status register that has a value of 0 in the mask sent with this command. All other event status register bits (bits which have a mask value of 1) are unaffected.

OptoScript Example

```
Status=TransmitReceiveString
(">ResetEventStatus,D,P1",ComHandle,ResponseString)
```

See Also

GetEventStatus (page 47)

SetAcceleration, GetAcceleration

buffered

Syntax

SetAcceleration, *axis, P1, P2*
GetAcceleration, *axis*

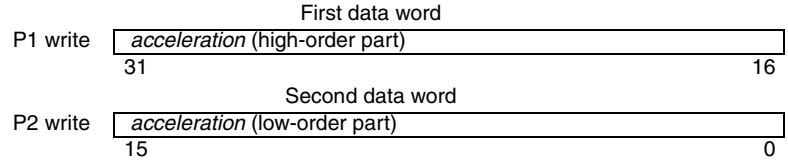
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

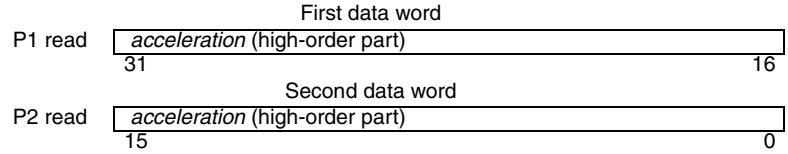
Name	Type	Range	Scaling	Units
acceleration	unsigned 32 bits	*0 to* 2^{31}-1	$1/2^{16}$	counts/cycle^2 microsteps/cycle^2

Data Structure

SetAcceleration



GetAcceleration



Description

SetAcceleration loads the maximum acceleration buffer register for the specified axis. This command is used with the Trapezoidal, Velocity Contouring, and S-curve profiling modes.

GetAcceleration reads the maximum acceleration buffer register.

Scaling example: To load a value of 1.750 counts/cycle^2 multiply by 65,536 (giving 114,688) and load the resultant number as a 32-bit number, giving 0001 in the high word and C000h in the low word. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Values returned by GetAcceleration must correspondingly be divided by 65,536 to convert to units of counts/cycle^2 or steps/cycle^2. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Restrictions

SetAcceleration may not be issued while an axis is in motion with the S-curve profile.

SetAcceleration is not valid in Electronic Gearing profile mode.

SetAcceleration is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

OptoScript Example

```
Status=TransmitReceiveString
(">SetAcceleration,C,P1,P2",ComHandle,ResponseString)
Status=TransmitReceiveString
(">GetAcceleration,C",ComHandle,ResponseString)
```

See Also SetDeceleration, GetDeceleration (page 80), SetJerk, GetJerk (page 88), SetPosition, GetPosition (page 95), SetVelocity, GetVelocity (page 109), MultiUpdate (page 55), Update (page 111)

SetActualPosition, GetActualPosition

Syntax

SetActualPosition, *axis*, *P1*, *P2*
GetActualPosition, *axis*

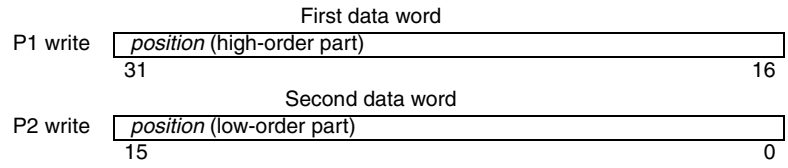
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

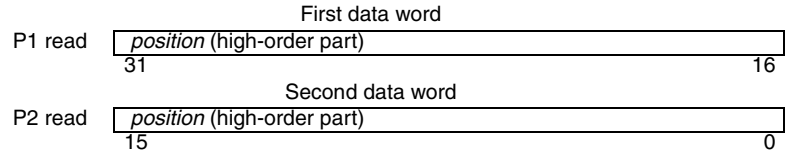
Name	Type	Range	Scaling	Units
position	signed 32 bits	2^{31} *to* 2^{31}-1	unity	counts microsteps

Data Structure

SetActualPosition



GetActualPosition



Description

SetActualPosition loads the position register (encoder position) for the specified axis. At the same time, the commanded position is replaced by the loaded value minus the position error. This prevents a servo "bump" when the new axis position is established. The destination position (see "SetPosition, GetPosition" on page 95) is also modified by this amount so that no trajectory motion will occur when the update instruction is issued. In effect, this instruction establishes a new reference position from which subsequent positions can be calculated. It is commonly used to set a known reference position after a homing procedure.

For axes configured as Pulse & Direction or Microstepping motor types, ActualPositionUnits determines if the position is specified and returned in units of counts or steps. Additionally, for these motor types, the position error is zeroed when the SetActualPosition command is sent. SetActualPosition takes effect immediately, it is not buffered. GetActualPosition reads the contents of the encoder's actual position register. This value will be accurate to within one cycle (as determined by Set/GetSampleTime).

OptoScript Example

```
Status=TransmitReceiveString
(">SetActualPosition,D,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetActualPosition,D",ComHandle,ResponseString)
```

See Also GetPositionError (page 51) SetActualPositionUnits, GetActualPositionUnits (page 63), AdjustActualPosition (page 37)

SetActualPositionUnits, GetActualPositionUnits

Syntax

SetActualPositionUnits, *axis, P1*
GetActualPositionUnits, *axis*

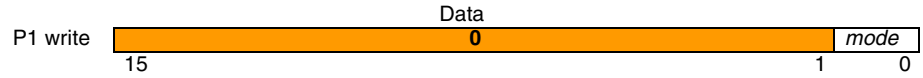
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

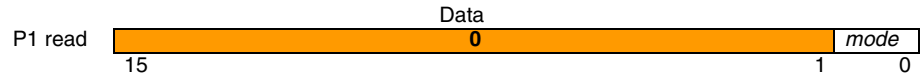
Name	Instance	Encoding
mode	Counts	0
	Steps	1

Data Structure

SetActualPositionUnits



GetActualPositionUnits



Description

SetActualPositionUnits determines the units used by the Set/GetActualPosition, AdjustActualPosition and GetCaptureValue for the specified axis. When set to Counts, position units are in encoder counts. When set to Steps, position units are in steps. The step position is calculated using the ratio as set by the SetEncoderToStepRatio command.

GetActualPositionUnits returns the position units for the specified axis.

OptoScript Example

```
Status=TransmitReceiveString
(">SetActualPositionUnits,E,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetActualPositionUnits,E",ComHandle,ResponseString)
```

See Also

SetActualPosition, GetActualPosition (page 61), SetEncoderToStepRatio, GetEncoderToStepRatio (page 83), AdjustActualPosition (page 37), GetCaptureValue (page 43)

SetAutoStopMode, GetAutoStopMode

Syntax

SetAutoStopMode, *axis, P1*
GetAutoStopMode, *axis*

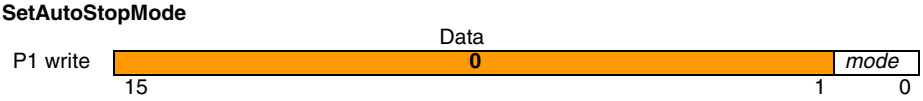
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

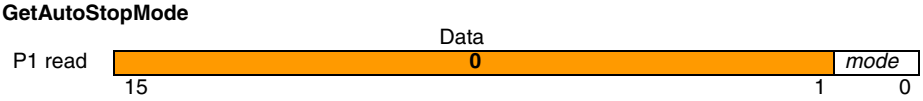
Name	Instance	Encoding
mode	Disable	0
	Enable	1

Data Structure

SetAutoStopMode



GetAutoStopMode



Description

SetAutoStopMode determines the behavior of the specified axis when a motion error occurs. When auto stop is enabled (SetAutoStopMode Enable), the axis goes into open-loop mode when a motion error occurs. When Auto-Stop is disabled (SetAutoStopMode Disable), the axis is not affected by a motion error.

GetAutoStopMode returns the state of the Auto-Stop mode.

Restrictions

When the encoder source is set to none (SetEncoderSource None), setting the auto stop mode to Enable will not stop motion in the event that the position error limit is exceeded.

OptoScript Example

```
Status=TransmitReceiveString
(">SetAutoStopMode,F,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetAutoStopMode,F",ComHandle,ResponseString)
```

See Also

GetEventStatus (page 47), SetPositionErrorLimit, GetPositionErrorLimit (page 96)

SetAxisMode, GetAxisMode

Syntax

SetAxisMode, *axis, P1*
GetAxisMode, *axis*

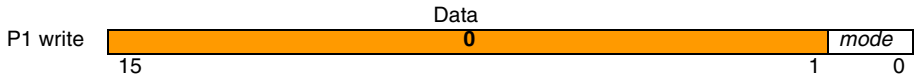
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

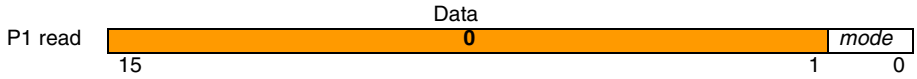
Name	Instance	Encoding
mode	Off	0
	On	1

Data Structure

SetAxisMode



GetAxisMode



Description

SetAxisMode enables (On) or disables (Off) the specified axis. A disabled axis will not respond to profile or other motion commands.

GetAxisMode returns the mode of the specified axis.

Restrictions

Disabled axes do not provide encoder feedback. If it is desired that an axis provide encoder feedback even though no profiling or servo control is to be used, the axis must be left enabled.

OptoScript Example

```
Status=TransmitReceiveString
(">SetAxisMode,0,P1",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetAxisMode,0",ComHandle,ResponseString)
```

SetAxisOutSource, GetAxisOutSource

Syntax

SetAxisOutSource, *axis, P1*
GetAxisOutSource, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

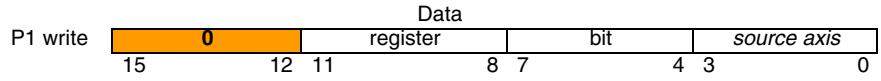
NOTE: The source axis must be on the same SNAP-SCM-BB4 as the axis argument.

Name	Type	Range	Board Address
sourceAxis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

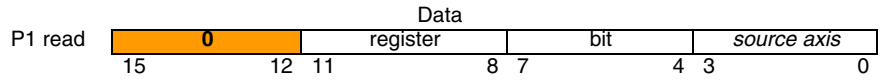
Name	Instance	Encoding
bit	*see below*	0 to 15
register	Disabled	0
	EventStatus	1
	ActivityStatus	
	SignalStatus	

Data Structure

SetAxisMode



GetAxisMode



Description

SetAxisOutSource maps the specified bit of the specified status register of axis to the AxisOut pin for the specified axis. The state of the AxisOut pin will thereafter track the state of bit. If register is absent (encoding of 0), bit is ignored, and the specified AxisOut pin is, in effect, turned off (inactive). When the AxisOutSource is set to disabled, the AxisOut signal can be set high or low using SetSignalSense bit 10.

GetAxisOutSource returns the mapping of the AxisOut pin of axis.

The following table shows the corresponding value for combinations of *bit* and *register*.

bit	event status register	activity status register	signal status register
0	Motion complete	Phasing initialized	Encoder A
1	Wrap-around	At maximum velocity	Encoder B

bit	event status register	activity status register	signal status register
2	Breakpoint 1	Tracking	Encoder index
3	Position capture		Home
4	Motion error		Positive limit
5	In positive limit		Negative limit
6	In negative limit		AxisIn
7	Instruction error	Axis settled	Hall sensor 1
8		Motor on/off	Hall sensor 2
9		Position capture	Hall sensor 3
0Ah		In motion	
0Bh	Commutation error	In positive limit	
0Ch		In negative limit	
0Dh			
0Eh	Breakpoint 2		
0Fh			

OptoScript Example

```
Status=TransmitReceiveString
(">SetAxisOutSource,1,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetAxisOutSource,1",ComHandle,ResponseString)
```

See Also SetSignalSense, GetSignalSense (page 102)

SetBreakPoint, GetBreakPoint

Syntax

SetBreakPoint, *axis, P1, P2*
GetBreakPoint, *axis, P1*

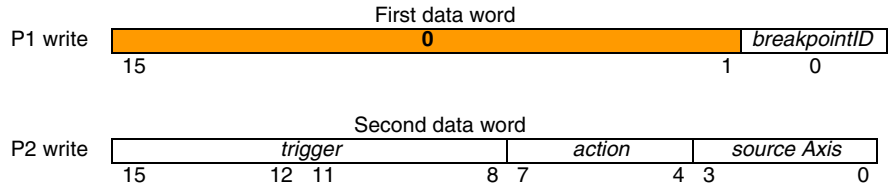
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

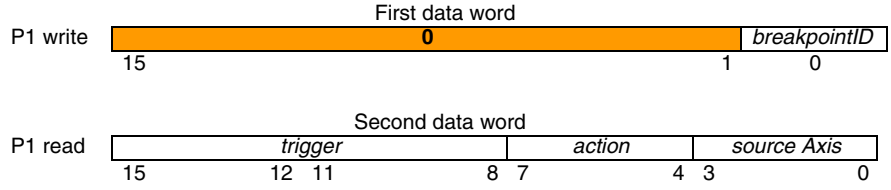
NOTE: The source axis must be on the same SNAP-SCM-BB4 as the axis argument.

Name	Instance	Encoding
breakpointID	Breakpoint1	0
	Breakpoint2	1
sourceAxis	Axis0	0
	Axis1	1
	Axis2	2
	Axis3	3
action	(none)	0
	Update	1
	AbruptStop	2
	SmoothStop	3
	MotorOff	4
trigger	(none)	0
	GreaterOrEqualCommandedPosition	1
	LesserOrEqualCommandedPosition	2
	GreaterOrEqualActualPosition	3
	LesserOrEqualActualPosition	4
	CommandedPositionCrossed	5
	ActualPositionCrossed	6
	Time	7
	EventStatus	8
	ActivityStatus	9
	SignalStatus	Ah

SetBreakPoint



GetBreakPoint



Description

SetBreakPoint establishes a breakpoint for the specified axis to be triggered by a condition or event on sourceAxis, which may be the same as or different from axis. Up to two concurrent breakpoints can be set for each axis, each of which may have its own breakpoint type and comparison value. The breakpointID field specifies which breakpoint the SetBreakPoint and GetBreakPoint commands will address.

The six Position breakpoints and the Time breakpoint are threshold-triggered; the breakpoint occurs when the indicated value reaches or crosses a threshold. The Status breakpoints are level-triggered; the breakpoint occurs when a specific bit or combination of bits in the indicated status register changes state. Thresholds and bit specifications are both set by the SetBreakPointValue instruction.

action determines what the motion processor does when the breakpoint occurs, as follows:

Action	Resultant command sequence
None	No action
Update	Update axis
AbruptStop	The profile executes an abrupt stop
SmoothStop	The profile executes a smooth stop
MotorOff	SetMotorMode axis, Off

GetBreakPoint returns the trigger, action, and axis for the specified breakpoint (0 or 1) of the indicated axis. When a breakpoint occurs, the trigger value will be reset to none. The CommandedPositionCrossed and the ActualPositionCrossed triggers are converted to one of the Position trigger types 1-4; depending on the current position when the command is issued.

Restrictions

Always load the breakpoint comparison value (SetBreakPointValue command) before setting a new breakpoint condition (SetBreakPoint command). Failure to do so will likely result in unexpected behavior.

OptoScript Example

```
Status=TransmitReceiveString
(">SetBreakPoint,2,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetBreakPoint,2,P1",ComHandle,ResponseString)
```

See Also

SetBreakPointValue, GetBreakPointValue (page 70)

SetBreakPointValue, GetBreakPointValue

Syntax

SetBreakPointValue, *axis, P1, P2, P3*
GetBreakPointValue, *axis, P1*

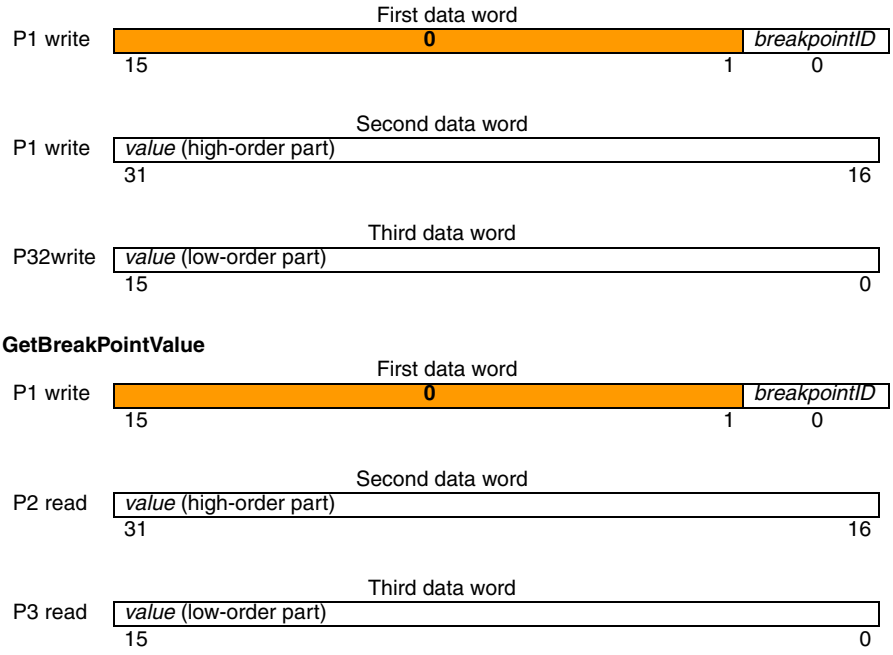
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
breakpointID	Breakpoint 1	0
	Breakpoint 2	1
value		*see below*

Data Structure

SetBreakPointValue



GetBreakPointValue

Description

SetBreakPointValue sets the breakpoint comparison value for the specified axis. For the position and time breakpoints, this is a threshold comparison value.

The value parameter is interpreted according to the trigger condition for the selected breakpoint; see "SetBreakPoint, GetBreakPoint" on page 68. The data format for each trigger condition is as follows:

Breakpoint Trigger	Value Type	Range	Units
GreaterOrEqualCommandedPosition	signed 32-bit	-2^{31} to $2^{31}-1$	counts

Breakpoint Trigger	Value Type	Range	Units
LesserOrEqualCommandedPosition	signed 32-bit	-2^{31} to $2^{31}-1$	counts
GreaterOrEqualActualPosition	signed 32-bit	-2^{31} to $2^{31}-1$	counts
LesserOrEqualActualPosition	signed 32-bit	-2^{31} to $2^{31}-1$	counts
CommandedPositionCrossed	signed 32-bit	-2^{31} to $2^{31}-1$	counts
ActualPositionCrossed	signed 32-bit	-2^{31} to $2^{31}-1$	counts
Time	unsigned 32-bit	0 to $2^{31}-1$	cycles
EventStatus	2 word mask	-	boolean status values
ActivityStatus	2 word mask	-	boolean status values
SignalStatus	2 word mask	-	boolean status values

For level-triggered breakpoints, the high-order part of value is the selection mask, and the low-order word is the sense mask. For each selection bit that is set to 1, the corresponding bit of the specified status register is conditioned to cause a breakpoint when it changes state. The sense-mask bit determines which state causes the break. If it is 1, the corresponding status-register bit will cause a break when it is set to 1. If it is 0, the status-register bit will cause a break when it is set to 0.

For example assume it is desired that the breakpoint type will be set to EventStatus and that a breakpoint should be recognized whenever the motion complete bit (bit 0 of event status register) is set to 1, or the commutation error bit (bit 11 of event status register) is set to 0. In this situation the high and low words for value would be high word: 0x801 (hex) and low word: 1.

GetBreakPointValue returns the breakpoint value for the specified breakpointID.

Two completely separate breakpoints are supported, each of which may have its own breakpoint type and comparison value. The breakpointID field specifies which breakpoint the SetBreakPointValue and GetBreakPointValue commands will address.

Restrictions

Always load the breakpoint comparison value (SetBreakPointValue command) before setting a new breakpoint condition (SetBreakPoint command). Failure to do so will likely result in unexpected behavior.

OptoScript Example

```
Status=TransmitReceiveString
(">SetBreakPointValue,3,P1,P2,P3",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetBreakPointValue,3,P1",ComHandle,ResponseString)
```

See Also

SetBreakPoint, GetBreakPoint (page 68)

SetBufferLength, GetBufferLength

Syntax

SetBufferLength, *axis, P1, P2, P3*
GetBufferLength, *axis, P1*

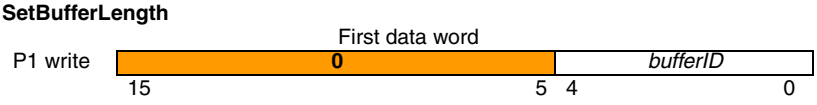
Arguments

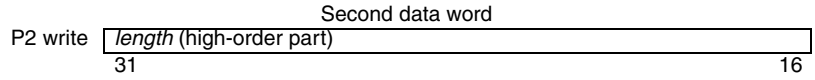
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

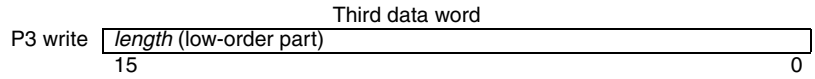
Name	Type	Range
bufferID	unsigned 16 bits	0 to 31
length	unsigned 32 bits	1 to 230 - 1

Data Structure

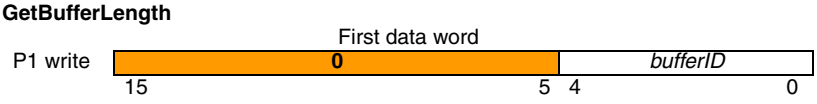
SetBufferLength

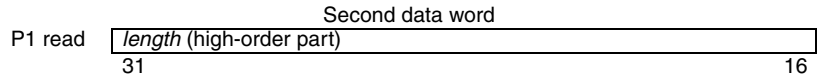


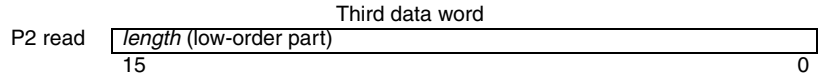




GetBufferLength







Description

SetBufferLength sets the length, in number of 32-bit elements, of the buffer in the memory block identified by bufferID.

NOTE: SetBufferLength resets the buffers read and write indexes to 0.

GetBufferLength returns the length of the specified buffer.

OptoScript Example

```
Status=TransmitReceiveString
(">SetBufferLength,4,P1,P2,P3",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetBufferLength,4,P1",ComHandle,ResponseString)
```

See Also

SetBufferReadIndex, GetBufferReadIndex (page 74), SetBufferStart, GetBufferStart (page 76), SetBufferWriteIndex, GetBufferWriteIndex (page 78)

SetBufferReadIndex, GetBufferReadIndex

Syntax

SetBufferReadIndex, *axis, P1, P2, P3*
GetBufferReadIndex, *axis, P1*

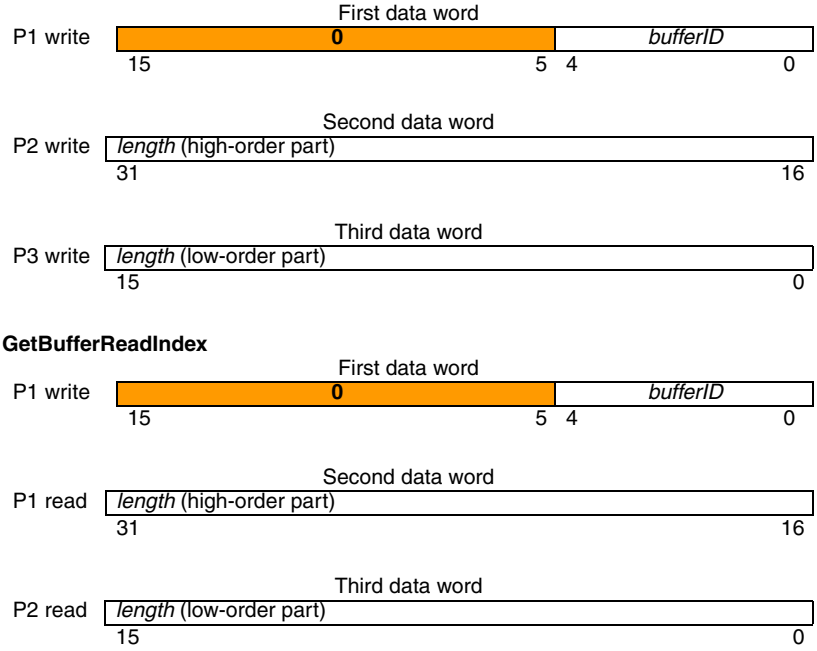
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
bufferID	unsigned 16 bits	0 *to* 31	unity	-
index	unsigned 32 bits	0 *to* buffer length - 1	unity	double words

Data Structure

SetBufferReadIndex



GetBufferReadIndex

Description

SetBufferReadIndex sets the address of the read index for the specified bufferID.

GetBufferReadIndex returns the current read index for the specified bufferID.

Restrictions

If the read index is set to an address beyond the length of the buffer, the command will not be executed and will return host I/O error code 7, buffer bound exceeded.

OptoScript Example

```
Status=TransmitReceiveString
(">SetBufferReadIndex,5,P1,P2,P3",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetBufferReadIndex,5,P1",ComHandle,ResponseString)
```

See Also

SetBufferLength, GetBufferLength (page 72), SetBufferStart, GetBufferStart (page 76), SetBufferWriteIndex, GetBufferWriteIndex (page 78)

SetBufferStart, GetBufferStart

Syntax

SetBufferStart, *axis, P1, P2, P3*
GetBufferStart, *axis, P1*

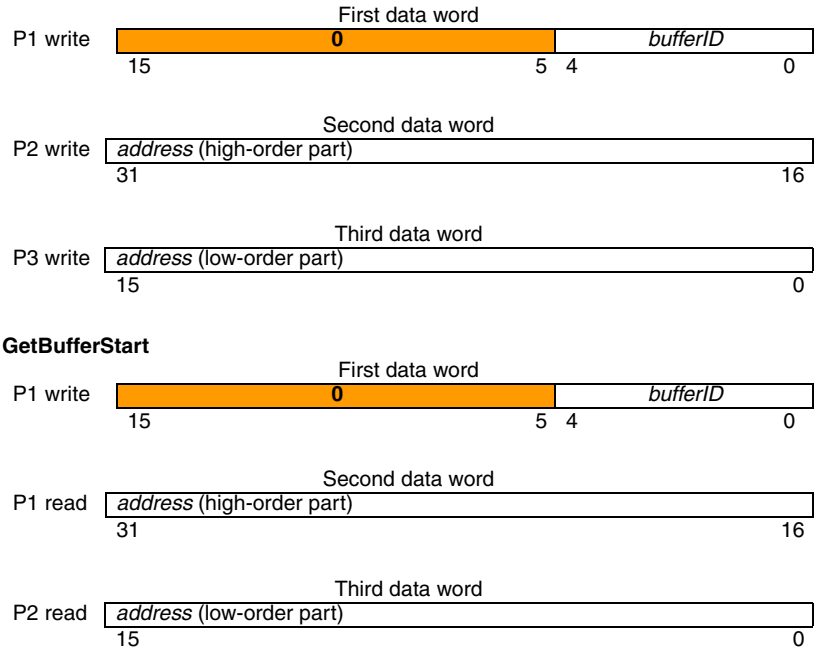
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
bufferID	unsigned 16 bits	0 *to* 31	unity	-
address	unsigned 32 bits	0 *to* 2^{31} - 1	unity	double words

Data Structure

SetBufferStart



GetBufferStart

Description

SetBufferStart sets the starting address for the specified buffer, in double-words, of the buffer in the memory block identified by bufferID.

NOTE: SetBufferStart resets the buffers read and write indexes to 0.

GetBufferStart returns the starting address for the specified bufferID.

OptoScript Example

```
Status=TransmitReceiveString
(">SetBufferStart,6,P1,P2,P3",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetBufferStart,6,P1",ComHandle,ResponseString)
```

See Also SetBufferLength, GetBufferLength (page 72), SetBufferReadIndex, GetBufferReadIndex (page 74), SetBufferWriteIndex, GetBufferWriteIndex (page 78)

SetBufferWriteIndex, GetBufferWriteIndex

Syntax

SetBufferWriteIndex, *axis, P1, P2, P3*
GetBufferWriteIndex, *axis, P1*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
bufferID	unsigned 16 bits	0 *to* 31	unity	-
index	unsigned 32 bits	0 *to* buffer length - 1	unity	double words

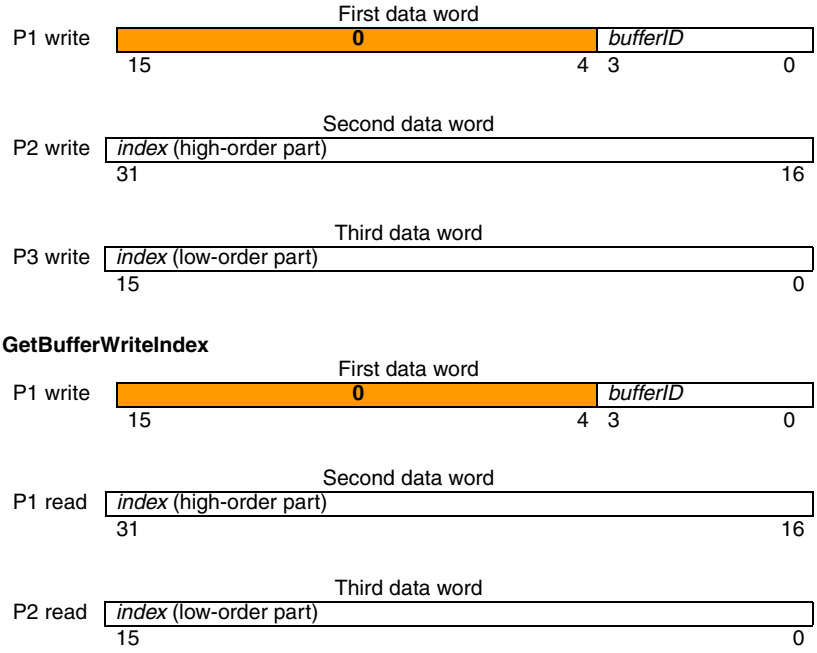
Description

SetBufferWriteIndex sets the write index for the specified bufferID.

GetBufferWriteIndex returns the write index for the specified bufferID.

Data Structure

SetBufferWriteIndex



GetBufferWriteIndex

OptoScript Example

```
Status=TransmitReceiveString
(">SetBufferWriteIndex,7,P1,P2,P3",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetBufferWriteIndex,7,P1",ComHandle,ResponseString)
```

See Also

SetBufferLength, GetBufferLength (page 72), SetBufferReadIndex, GetBufferReadIndex (page 74), SetBufferStart, GetBufferStart (page 76)

SetCaptureSource, GetCaptureSource

Syntax

SetCaptureSource, *axis*, *P1*
GetCaptureSource, *axis*

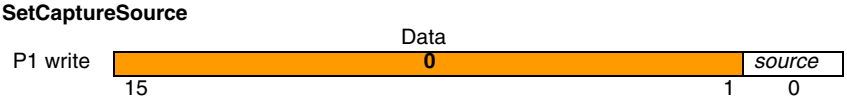
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

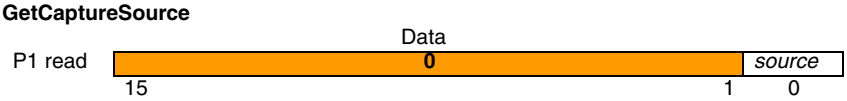
Name	Instance	Encoding
source	Index	0
	Home	1

Data Structure

SetCaptureSource



GetCaptureSource



Description

SetCaptureSource determines which of two encoder signals, Index or Home, is used to trigger the high-speed capture of the actual axis position for the specified axis.

GetCaptureSource returns the capture signal source for the selected axis.

OptoScript Example

```
Status=TransmitReceiveString
(">SetCaptureSource,2,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetCaptureSource,2",ComHandle,ResponseString)
```

See Also

GetCaptureValue (page 43)

SetDeceleration, GetDeceleration

buffered

Syntax

SetDeceleration, *axis, P1, P2*
GetDeceleration, *axis*

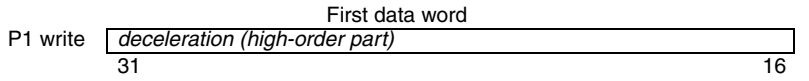
Arguments

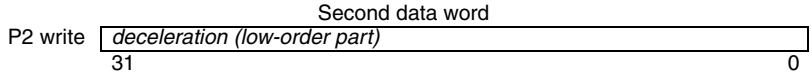
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
deceleration	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{16}$	counts/cycle^2 microsteps/cycle^2

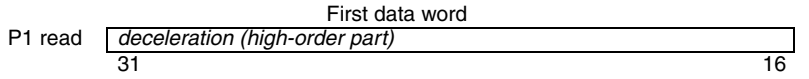
Data Structure

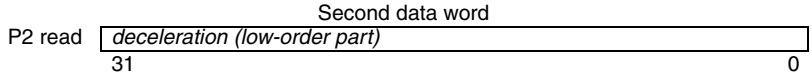
SetDeceleration





GetDeceleration





Description

SetDeceleration loads the maximum deceleration buffer register for the specified axis.

GetDeceleration returns the value of the maximum deceleration buffer.

Scaling example: To load a value of 1.750 counts/cycle2 multiply by 65,536 (giving 114,688) and load the resultant number as a 32-bit number, giving 0001 in the high word and C000h in the low word. Retrieved numbers (GetDeceleration) must correspondingly be divided by 65,536 to convert to units of counts/cycle^2 or steps/cycle^2 . See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Restrictions

This is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction is entered.

These commands are used with the Trapezoidal and Velocity Contouring profile modes. They are not used with the Electronic Gearing or S-curve profile mode.

NOTE: If deceleration is set to zero, then the value specified for acceleration (SetAcceleration) will automatically be used to set the magnitude of deceleration.

OptoScript Example

```
Status=TransmitReceiveString
(">SetDeceleration,5,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetDeceleration,5",ComHandle,ResponseString)
```

See Also

SetAcceleration, GetAcceleration (page 59), SetPosition, GetPosition (page 95), SetVelocity, GetVelocity (page 109), MultiUpdate (page 55), Update (page 111)

SetEncoderSource, GetEncoderSource

Syntax

SetEncoderSource, *axis*, *P1*
GetEncoderSource, *axis*

Arguments

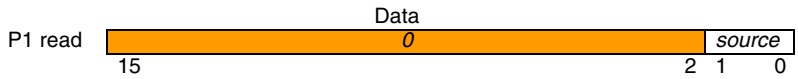
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
source	Incremental	0
	reserved	1
	None	3
	Loopback	4

Data Structure

SetEncoderSource

GetEncoderSource



Description

SetEncoderSource sets the type of feedback (incremental quadrature encoder or parallel-word) for the specified axis. When incremental quadrature is selected the motion processor expects A and B quadrature signals to be input at the quad A/B axis inputs. 1When motor type (see page 94) is set to Pulse & Direction and the encoder source is set to Loopback, the step output is internally fed back into the quad counters. This allows for position capture of the step position when a physical encoder is not present.

GetEncoderSource returns the code for the current type of feedback.

Restrictions

A Loopback source is only supported for Pulse & Direction motors. A source value of None is only valid for Microstepping and Pulse & Direction motors.

OptoScript Example

```
Status=TransmitReceiveString
("">SetEncoderSource,7,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
("">GetEncoderSource,7",ComHandle,ResponseString)
```

SetEncoderToStepRatio, GetEncoderToStepRatio

Syntax

SetEncoderToStepRatio, *axis*, *P1*, *P2*
GetEncoderToStepRatio, *axis*

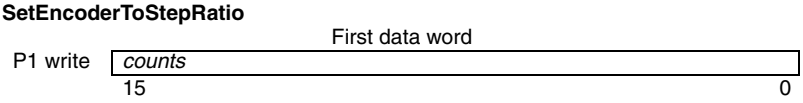
Arguments

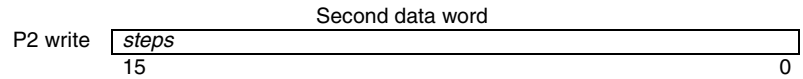
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
counts	unsigned 16 bits	0 *to* 2^{15}-1	unity	counts
steps	unsigned 16 bits	0 *to* 2^{15}-1	unity	microsteps

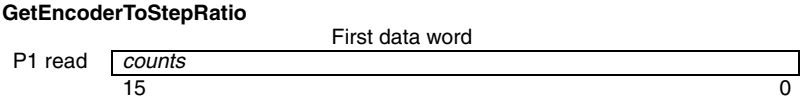
Data Structure

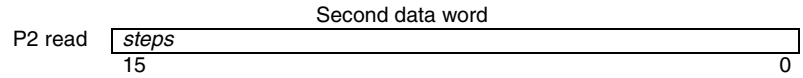
SetEncoderToStepRatio





GetEncoderToStepRatio





Description

SetEncoderToStepRatio sets the ratio of the number of encoder counts to the number of output steps per motor rotation used by the motion processor to convert encoder counts into steps. Counts is the number of encoder counts per full rotation of the motor. Steps is the number of steps output by the motion processor per full rotation of the motor. Since this command sets a ratio, the parameters do not have to be for a full rotation as long as they correctly represent the encoder count to step ratio.

GetEncoderToStepRatio returns the ratio of the number of encoder counts to the number of output steps per motor rotation.

OptoScript Example

```
Status=TransmitReceiveString
(">SetEncoderToStepRatio,8,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetEncoderToStepRatio,8",ComHandle,ResponseString)
```

See Also

SetActualPositionUnits, GetActualPositionUnits (page 63)

SetGearMaster, GetGearMaster

Syntax

SetGearMaster, *axis, P1*
GetGearMaster, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

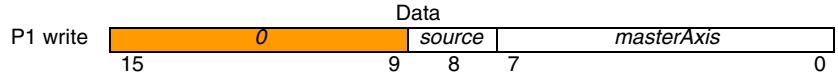
NOTE: The master axis must be on the same SNAP-SCM-BB4 as the axis argument.

Name	Type	Range	Board Address
masterAxis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

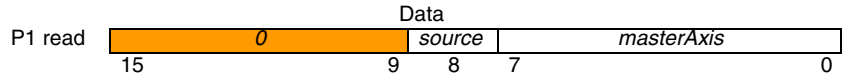
Name	Instance	Encoding
source	Actual	0
	Commanded	1

Data Structure

SetGearMaster



GetGearMaster



Description

SetGearMaster establishes the slave (axis) and master (masterAxis) axes for the electronic-gearing profile, and sets the source, Actual or Commanded, of the master axis position data to be used.

The masterAxis determines the axis that will drive the slave axis. Both the slave and the master axes must be enabled (SetAxisMode command). The source determines whether the master axis' commanded position as determined by the trajectory generator will be used to drive the slave axis, or whether the master axis' encoder position will be used to drive the slave.

GetGearMaster returns the value for the geared axes and position source.

OptoScript Example

```
Status=TransmitReceiveString
(">SetGearMaster,A,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetGearMaster,A",ComHandle,ResponseString)
```

See Also

SetGearRatio, GetGearRatio (page 85)

SetGearRatio, GetGearRatio

buffered

Syntax

SetGearRatio, *slaveAxis, P1, P2*
GetGearRatio, *slaveAxis*

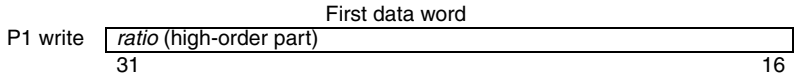
Arguments

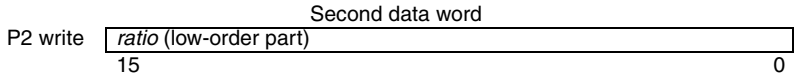
	Type	Range	Board Address
slaveaxis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
ratio	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	SlaveCts/MasterCts

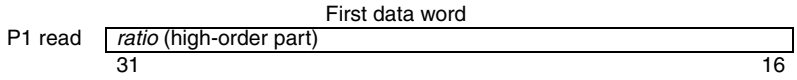
Data Structure

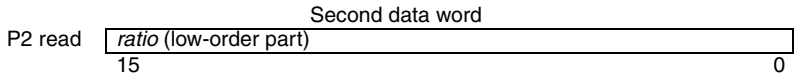
SetGearRatio





GetGearRatio





Description

SetGearRatio sets the ratio between the master and slave axes for the electronic gearing profile for the current axis. Positive ratios cause the slave to move in the same direction as the master, negative ratios in the opposite direction. The specified ratio has a unity scaling of 65,536.

GetGearRatio returns the gear ratio set for the specified slave axis.

Scaling examples:

Ratio Value	Resultant Ratio
-32,768	.5 negative slave counts for each positive master count
1,000,000	15.259 positive slave counts for each positive master count
123 .0018	positive slave counts for each positive master count

Restrictions

This is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction is entered.

OptoScript Example

```
Status=TransmitReceiveString
(">SetGearRatio,B,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetGearRatio,B",ComHandle,ResponseString)
```

See Also SetGearMaster, GetGearMaster (page 84), MultiUpdate (page 55), Update (page 111)

SetInterruptMask, GetInterruptMask

Syntax

SetInterruptMask, *axis, P1*
GetInterruptMask, *axis*

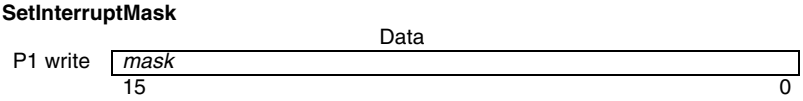
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

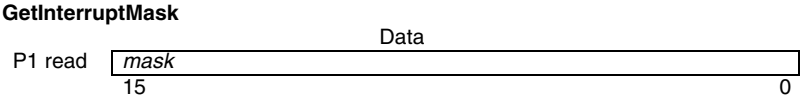
Name	Instance	Encoding
mask	Motion complete	0001h
	Wrap-around	0002h
	Breakpoint 1	0004h
	Capture received	0008h
	Motion error	0010h
	In positive limit	0020h
	In negative limit	0040h
	Instruction error	0080h
	Commutation error	0800h
	Breakpoint 2	4000h

Data Structure

SetInterruptMask



GetInterruptMask



Description

SetInterruptMask determines which bits in the event status register of the specified axis will cause a host interrupt. For each interrupt mask bit that is set to 1, the corresponding event status register bit will cause an interrupt when that status register bit goes active (is set to 1). Interrupt mask bits set to 0 will not generate interrupts.

GetInterruptMask returns the mask for the specified axis.

Example: The interrupt mask value 28h will generate an interrupt when either the "in positive limit" bit or the "capture received" bit of the event status register goes active (set to 1).

OptoScript Example

```
Status=TransmitReceiveString
(">SetInterruptMask,C,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetInterruptMask,C",ComHandle,ResponseString)
```

See Also

ClearInterrupt (page 38), GetInterruptAxis (page 50)

SetJerk, GetJerk

buffered

Syntax

SetJerk, *axis, P1, P2*
GetJerk, *axis*

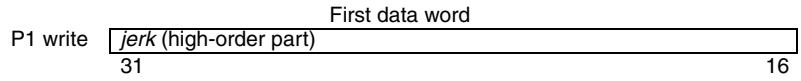
Arguments

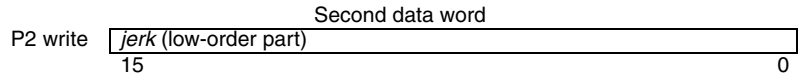
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
jerk	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{32}$	counts/cycle^3 microsteps/cycle^3

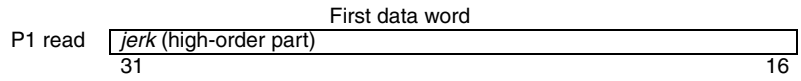
Data Structure

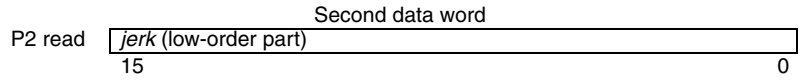
SetJerk





GetJerk





Description

SetJerk loads the jerk register in the parameter buffer for the specified axis.

GetJerk reads the contents of the Jerk register.
Scaling example: To load a jerk value (rate of change of acceleration) of 0.012345 counts/cycle3 (or steps/cycle3) multiply by 232 or 4,294,967,296. In this example this gives a value to load of 53,021,371 (decimal) which corresponds to a high word of 0329h and a low word of 0ABBh when loading each word in hexadecimal. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Restrictions

SetJerk is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

This command is used only with the S-curve profile mode. It is not used with the trapezoidal, velocity contouring, or electronic gear profile modes.

OptoScript Example

```
Status=TransmitReceiveString
(">SetJerk,D,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetJerk,D",ComHandle,ResponseString)
```

See Also

SetAcceleration, GetAcceleration (page 59), SetDeceleration, GetDeceleration (page 80), SetPosition, GetPosition (page 95), SetVelocity, GetVelocity (page 109), MultiUpdate (page 55), Update (page 111)

SetLimitSwitchMode, GetLimitSwitchMode

Syntax

SetLimitSwitchMode, *axis*, *P1*
GetLimitSwitchMode, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
mode	Off	0
	On	1

Data Structure

SetLimitSwitchMode

GetLimitSwitchMode

Description

SetLimitSwitchMode enables (On) or disables (Off) limit-switch sensing for the specified axis. When the mode is enabled, the axis will cause the corresponding limit-switch bits in the event status register and activity status register to be set when it enters either the positive or negative limit switches and the axis will be brought to an abrupt stop. When it is disabled, the status bits are not set, and the axis is not stopped, regardless of whether or not the axis is in a limit switch.

GetLimitSwitchMode returns the value for the state of the limit-sensing mode.

OptoScript Example

```
Status=TransmitReceiveString
(">SetLimitSwitchMode,1,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetLimitSwitchMode,1",ComHandle,ResponseString)
```

See Also

GetActivityStatus (page 40), GetEventStatus (page 47)

SetMotionCompleteMode, GetMotionCompleteMode

Syntax

SetMotionCompleteMode, *axis, P1*
GetMotionCompleteMode, *axis*

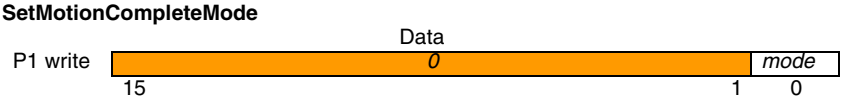
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

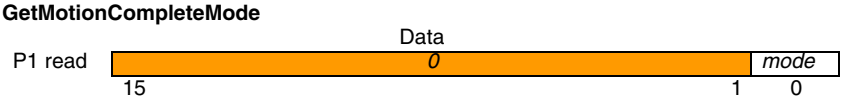
Name	Instance	Encoding
mode	commanded	0
	actual	1

Data Structure

SetMotionCompleteMode



GetMotionCompleteMode



Description

SetMotionCompleteMode establishes the source for the comparison which determines the motion complete status for the specified axis. When set to commanded mode the motion is considered complete when the profile velocity reaches zero and no further motion will occur without an additional host command. This mode is unaffected by the actual encoder location.

When set to actual mode the motion complete bit will be set when the above condition is true, and when the actual encoder position has been within the settle window (SetSettleWindow command) for the number of cycles specified by the SetSettleTime command. The settle timer is started at zero at the end of the trajectory profile motion, so at a minimum a delay of SettleTime cycles will occur after the trajectory profile motion is complete.

GetMotionCompleteMode returns the value for the motion-complete mode.

OptoScript Example

```
Status=TransmitReceiveString
(">SetMotionCompleteMode,2,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetMotionCompleteMode,2",ComHandle,ResponseString)
```

See Also

SetSettleTime, GetSettleTime (page 100), SetSettleWindow, GetSettleWindow (page 101)

SetMotorMode, GetMotorMode

Syntax

SetMotorMode, *axis, P1*
GetMotorMode, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
mode	Off	0
	On	1

Data Structure

SetMotorMode

GetMotorMode

Description

SetMotorMode determines the mode of motor operation. When set to On, several events take place. For step motor and microstepping axes, the trajectory generator controls the motor output. For all motor types, when the encoder source (Set/GetEncoderSource) is set to incremental or parallel, the position error is cleared; equivalent to a ClearPositionError command.

When the motor mode is set to Off, the axis is in open-loop mode, and is controlled by commands placed directly into the motor output register by the host. Setting the motor mode to Off also resets the trajectory generator, bringing any active motion to an abrupt stop. In addition, the maximum velocity (Set/GetVelocity) is set to zero. On axes configured for step motor and microstepping motor types, the step generator is switched off when the motor mode is set to Off. The following table shows the motor output source for each motor type and mode.

Motor type	Motor mode	Motor output source
Pulse & direction;	Off	N/A
microstepping	On	Trajectory generator

GetMotorMode returns the value of the motor mode.

OptoScript Example

```
Status=TransmitReceiveString
(">SetMotorMode,3,P1",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetMotorMode,3",ComHandle,ResponseString)
```

See Also GetActivityStatus (page 40)

SetMotorType, GetMotorType

Syntax

SetMotorType, *axis, P1*
GetMotorType, *axis*

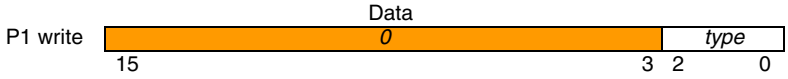
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

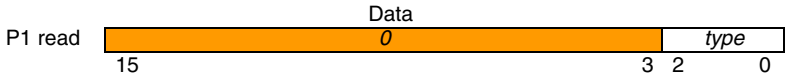
Name	Instance	Encoding
type	reserved	0
	reserved	1
	reserved	2
	reserved	3
	reserved	4
	reserved	7

Data Structure

SetMotorMode



GetMotorMode



Description

SetMotorType sets type of motor being driven by the selected axis. This operation sets the number of phases for commutation on the axis, as well as internally configuring the motion processor for the motor type.

Each motor type, and the number of phases to be commutated, are detailed in the following table.

Motor type	Commutation
Pulse & Direction	None

Should this be removed?

GetMotorType returns the configured motor type for the selected axis.

Restrictions

The motor type should only be set once for each axis; either via the motor configuration word during device startup, or immediately after reset using SetMotorType. Once it has been set, it should not be changed.

OptoScript Example

```
Status=TransmitReceiveString
(">SetMotorType,4,P1",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetMotorType,4",ComHandle,ResponseString)
```

SetPosition, GetPosition

buffered

Syntax

SetPosition, *axis, P1, P2*
GetPosition, *axis*

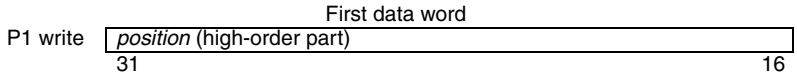
Arguments

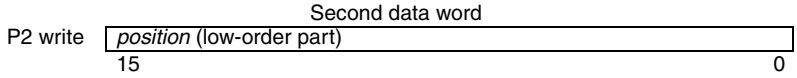
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
position	signed 32 bits	-2^{31} *to* $2^{31}-1$	unity	counts microsteps

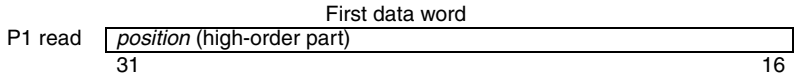
Data Structure

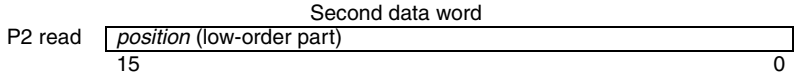
SetPosition





GetPosition





Description

SetPosition specifies the trajectory destination of the specified axis. It is used in the Trapezoidal and S-curve profile modes.

GetPosition reads the contents of the buffered position register.

Restrictions

SetPosition is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

OptoScript Example

```
Status=TransmitReceiveString
(">SetPosition,5,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetPosition,5",ComHandle,ResponseString)
```

See Also

SetAcceleration, GetAcceleration (page 59), SetDeceleration, GetDeceleration (page 80), SetJerk, GetJerk (page 88), SetVelocity, GetVelocity (page 109), MultiUpdate (page 55), Update (page 111)

SetPositionErrorLimit, GetPositionErrorLimit

Syntax

SetPositionErrorLimit, *axis*, *P1*, *P2*
GetPositionErrorLimit, *axis*

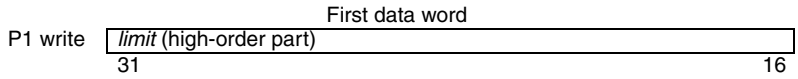
Arguments

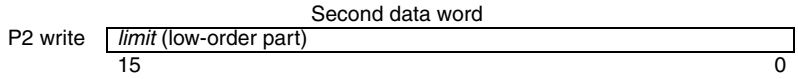
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
limit	unsigned 32 bits	0 *to* 2^{31}-1	unity	counts

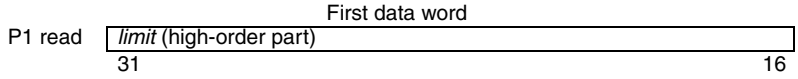
Data Structure

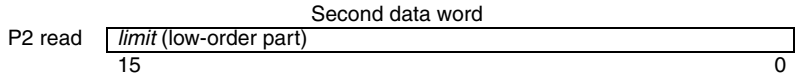
SetPositionErrorLimit





GetPositionErrorLimit





Description

SetPositionErrorLimit sets the absolute value of the maximum position error allowable by the motion processor for the specified axis. If the position error exceeds this limit, a motion error occurs. Such a motion error may or may not cause the axis to stop moving depending on the value set using the SetAutoStopMode command.

GetPositionErrorLimit returns the value of the position error limit.

OptoScript Example

```
Status=TransmitReceiveString
("">SetPositionErrorLimit,6,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString
("">GetPositionErrorLimit,6",ComHandle,ResponseString)
```

See Also

GetPositionError (page 51), SetActualPosition, GetActualPosition (page 61), SetPosition, GetPosition (page 95)

SetProfileMode, GetProfileMode

buffered

Syntax

SetProfileMode, *axis, P1*
GetProfileMode, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Instance	Encoding
profile	Trapezoidal	0
	Velocity contouring	1
	S-curve	2
	Electronic gear	3

Data Structure

SetProfileMode

GetProfileMode

Description

SetProfileMode sets the profile mode for the specified axis.

GetProfileMode returns the contents of the buffered profile-mode register for the specified axis.

Restrictions

SetProfileMode is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

OptoScript Example

```
Status=TransmitReceiveString
(">SetProfileMode,7,P1",ComHandle,ResponseString)
```

See Also

MultiUpdate (page 55), Update (page 111)

SetSampleTime, GetSampleTime

Syntax

SetSampleTime, *axis*, *P1*, *P2*
GetSampleTime, *axis*

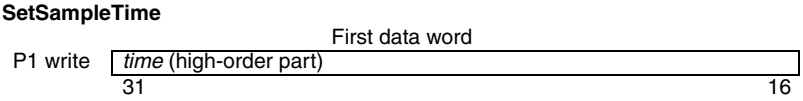
Arguments

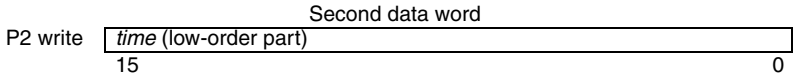
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Units
time	unsigned 32 bits	51 *to* 2^{20}	microseconds

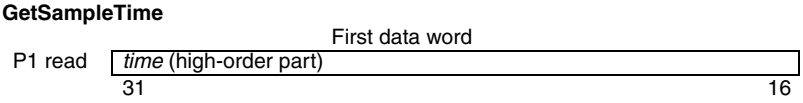
Data Structure

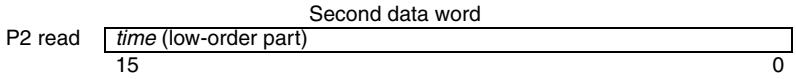
SetSampleTime





GetSampleTime





Description

SetSampleTime sets the time basis for the motion processor. This time basis determines the trajectory update rate for all motor types as well.

The time value is expressed in microseconds. The motion processor hardware can adjust the cycle time only in increments of 51.2 microseconds; the time value passed to this command will be rounded up to the nearest increment of this base value.

Minimum cycle time depends on the number of enabled axes as follows:

# enabled axes	minimum cycle time	cycle time w/ trace capture	time per axis	maximum cycle frequency
1	51.2 µs	102.4 µs	51.2 µs / 102.4 µs	19.53 KHz (9.76 w/ trace capture)
2	153.6 µs	153.6 µs	76.8 µs	6.51 KHz
3	204.8 µs	204.8 µs	68.3 µs	4.88 KHz
4	256 µs	256 µs	64 µs	3.91 KHz

Using the trace feature on single axis products with the sample time set to 51.2µs will result in unexpected behavior.

GetSampleTime returns the value of the sample time.

Restrictions

This command affects the cycle time for all axes on a given SNAP-SCM-BB4.

This command cannot be used to set a sample time lower than the required minimum cycle time for the current configuration. Attempting to do so will set the sample time to the required minimum cycle time as specified in the previous table.

OptoScript Example

```
Status=TransmitReceiveString
(">SetSampleTime,P1,P2,ComHandle,ResponseString)

Status=TransmitReceiveString (">GetSampleTime,ComHandle,ResponseString)
```

SetSettleTime, GetSettleTime

Syntax

SetSettleTime, *axis, P1*
GetSettleTime, *axis*

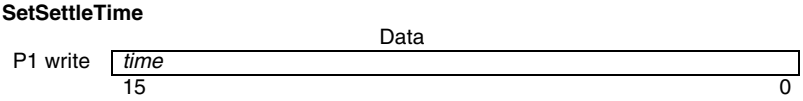
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

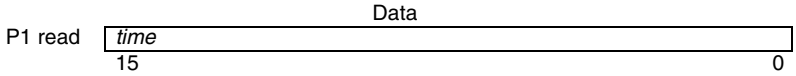
Name	Type	Range	Scaling	Units
time	unsigned 16 bits	0 *to* 2^{15}-1	unity	cycles

Data Structure

SetSettleTime



GetSettleTime



Description

SetSettleTime sets the time, in number of cycles, that the specified axis must remain within the settle window before the axis-settled indicator in the activity status register is set.

GetSettleTime returns the value of the settle time for the specified axis.

OptoScript Example

```
Status=TransmitReceiveString
(">SetSettleTime,8,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetSettleTime,8",ComHandle,ResponseString)
```

See Also

SetMotionCompleteMode, GetMotionCompleteMode (page 91), SetSettleWindow, GetSettleWindow (page 101), GetActivityStatus (page 40)

SetSettleWindow, GetSettleWindow

Syntax

SetSettleWindow, *axis, P1*
GetSettleWindow, *axis*

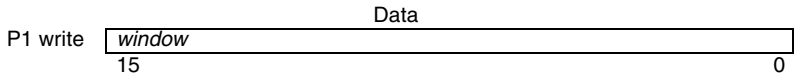
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

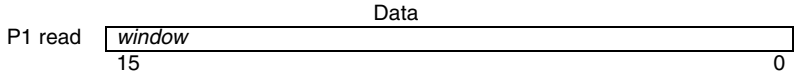
Name	Type	Range	Scaling	Units
window	unsigned 16 bits	0 *to* 2^{16}-1	unity	cycles

Data Structure

SetSettleWindow



GetSettleWindow



Description

SetSettleWindow sets the position range within which the specified axis must remain for the duration specified by SetSettleTime before the axis-settled indicator in the activity status register is set.

GetSettleWindow returns the value of the settle window.

OptoScript Example

```
Status=TransmitReceiveString
(">SetSettleWindow,A,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetSettleWindow,A",ComHandle,ResponseString)
```

See Also

SetMotionCompleteMode, GetMotionCompleteMode (page 91), SetSettleTime, GetSettleTime (page 100), GetActivityStatus (page 40)

SetSignalSense, GetSignalSense

Syntax

SetSignalSense, *axis, P1*
GetSignalSense, *axis*

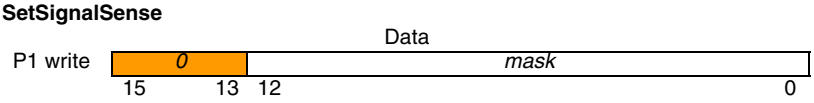
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

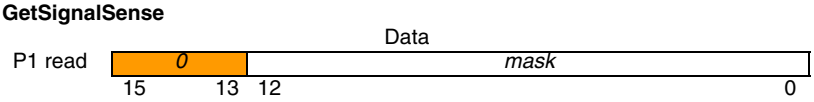
Name	Indicator	Encoding	Bit Number
mask	EncoderA	0001h	0
	EncoderB	0002h	1
	EncoderIndex	0004h	2
	EncoderHome	0008h	3
	PositiveLimit	0010h	4
	NegativeLimit	0020h	5
	AxisIn	0040h	6
	reserved	0080h	7
	reserved	0100h	8
	reserved	0200h	9
	AxisOut	0400h	10
	StepOutput	0800h	11
	MotorDirection	1000h	12
	reserved		13 - 15

Data Structure

SetSignalSense



GetSignalSense



Description

SetSignalSense establishes the sense of the corresponding bits of the signal status register, with the addition of StepOutput and MotorDirection, for the specified axis. For all input signals, the input is inverted if the corresponding sense bit is one; otherwise it is not inverted.

For encoder index/home: if the sense bit is 1, a capture will occur on a low-to-high signal transition. Otherwise, a capture will occur on a high-to-low transition.

For positive and negative limit: if the sense bit is 1, an over-travel condition will occur if the signal is high. Otherwise, an over-travel condition will occur when the signal is low.

The AxisOut signal is inverted if the sense bit is set to one; otherwise it is not inverted.

When the StepOutput bit is set to 1, a step will be generated by the motion processor with a low-to-high transition on the Pulse signal. Otherwise, a step will be generated by the motion processor with a high-to-low transition on the Pulse signal.

Setting the MotorDirection bit has the effect of swapping the sense of positive and negative motor movement.

GetSignalSense returns the value of the signal sense mask.

Restrictions

Inverting the encoder A or B signal may prevent the index capture mechanism from operating correctly.

OptoScript Example

```
Status=TransmitReceiveString
(">SetSignalSense,B,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetSignalSense,B",ComHandle,ResponseString)
```

See Also

GetSignalStatus (page 52)

SetStartVelocity, GetStartVelocity

Syntax

SetStartVelocity, *axis*, *P1*, *P2*
GetStartVelocity, *axis*

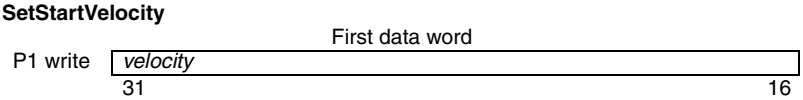
Arguments

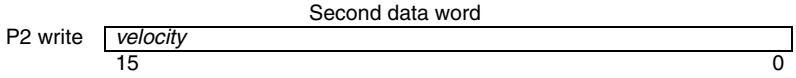
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
velocity	unsigned 32 bits	0 *to* 2^{31}-1	$1/2^{16}$	counts/cycle microsteps/cycle

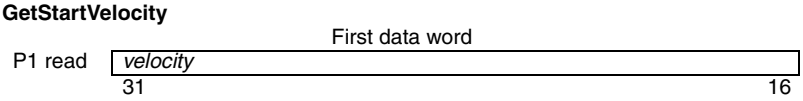
Data Structure

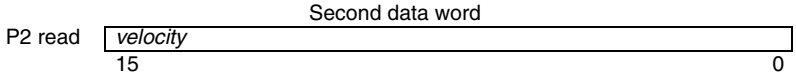
SetStartVelocity





GetStartVelocity





Description

SetStartVelocity loads the starting velocity register for the specified axis. The start velocity is the instantaneous velocity at the start and at the end of the profile.

GetStartVelocity reads the value of the starting velocity register.
Scaling example: To load a starting velocity value of 1.750 counts/cycle multiply by 65,536 (giving 114,688) and load the resultant number as a 32-bit number, giving 0001 in the high word and C000h in the low word. Values returned by GetStartVelocity must correspondingly be divided by 65,536 to convert them to units of counts/cycle. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Restrictions

StartVelocity is only used in the Velocity Contouring and Trapezoidal profile modes.

OptoScript Example

```
Status=TransmitReceiveString
(">SetStartVelocity,C,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetStartVelocity,C,P1",ComHandle,ResponseString)
```

See Also SetVelocity, GetVelocity (page 109), SetAcceleration, GetAcceleration (page 59), SetDeceleration, GetDeceleration (page 80), SetPosition, GetPosition (page 95)

SetStepRange, GetStepRange

Syntax

SetStepRange, *axis, P1*
GetStepRange, *axis*

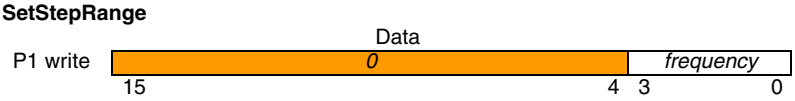
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

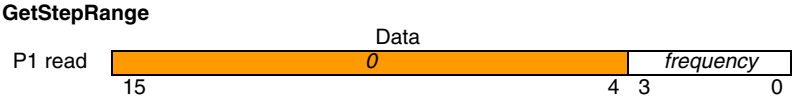
Name	Instance	Encoding
frequency	5 MHz	1
	625 kHz	4
	156.25 kHz	6
	39.062 kHz	8

Data Structure

SetStepRange



GetStepRange



Description

SetStepRange sets the maximum pulse rate frequency for the specified axis. For example, if the desired maximum pulse rate is 200,000 pulses/second, the command SetStepRange 4 should be issued.

GetStepRange returns the maximum pulse rate frequency for the specified axis.

OptoScript Example

```
Status=TransmitReceiveString
(">SetStepRange,D,P1",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetStepRange,D",ComHandle,ResponseString)
```

SetStopMode, GetStopMode

buffered

Syntax

SetStopMode, *axis, P1*
GetStopMode, *axis*

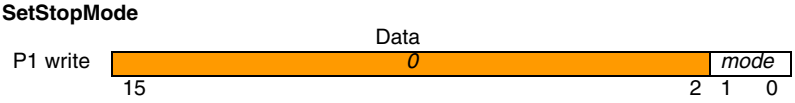
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

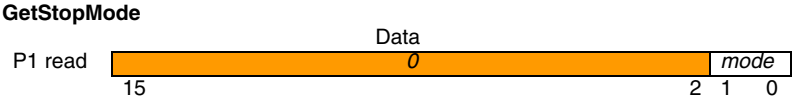
Name	Instance	Encoding
mode	Disabled	0
	AbruptStop	1
	SmoothStop	2

Data Structure

SetStopMode



GetStopMode



Description

SetStopMode stops the specified axis. The available stop modes are AbruptStop, which instantly (without any deceleration phase) stops the axis, SmoothStop which uses the programmed deceleration value and profile shape for the current profile mode to stop the axis, or Disabled, which is generally used to turn off a previously issued set stop command.

NOTE: After an Update, a buffered stop command (SetStopMode command) will reset to the Disabled condition. In other words, if the command SetStopMode is followed by an Update command and then by a GetStopMode command, the retrieved stop mode will be Disabled.

GetStopMode returns the value of the stop mode.

Restrictions

SmoothStop mode is not available in the Electronic gear profile mode.

SetStopMode is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

OptoScript Example

```
Status=TransmitReceiveString
(">SetStopMode,E,P1",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetStopMode,E",ComHandle,ResponseString)
```

See Also

MultiUpdate (page 55), Update (page 111)

SetTrackingWindow, GetTrackingWindow

Syntax

SetTrackingWindow, *axis, P1*
GetTrackingWindow, *axis*

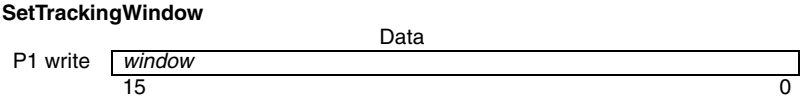
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

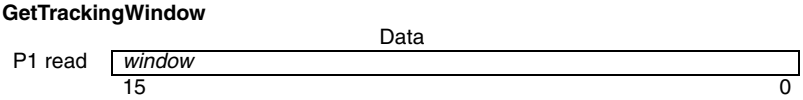
Name	Type	Range	Scaling	Units
window	unsigned 16 bits	0 *to* 2^{16}-1	unity	counts

Data Structure

SetTrackingWindow



GetTrackingWindow



Description

SetTrackingWindow sets boundaries for the position error of the specified axis. If the absolute value of the position error exceeds the tracking window, the tracking indicator (bit 2 of the activity status register) is set to 0. When the position error returns to within the window, the tracking indicator is set to 1.

GetTrackingWindow returns the value of the tracking window.

OptoScript Example

```
Status=TransmitReceiveString
(">SetTrackingWindow,F,P1",ComHandle,ResponseString)

Status=TransmitReceiveString
(">GetTrackingWindow,F",ComHandle,ResponseString)
```

See Also

GetActivityStatus (page 40), SetActualPosition, GetActualPosition (page 61)

SetVelocity, GetVelocity

buffered

Syntax

SetVelocity, *axis, P1, P2*
GetVelocity, *axis,*

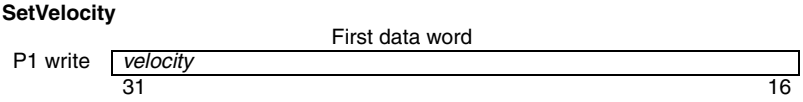
Arguments

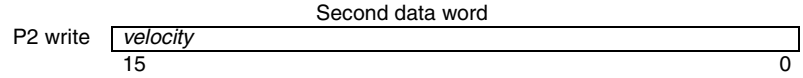
	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range	Scaling	Units
velocity	signed 32 bits	-2^{31} *to* $2^{31}-1$	$1/2^{16}$	counts/cycle microsteps/cycle

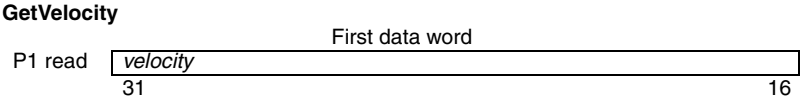
Data Structure

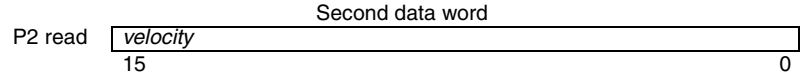
SetVelocity





GetVelocity





Description

SetVelocity loads the maximum velocity buffer register for the specified axis.

GetVelocity returns the contents of the maximum velocity buffer register.
Scaling example: To load a velocity value of 1.750 counts/cycle, multiply by 65,536 (giving 114,688) and load the resultant number as a 32-bit number; giving 0001 in the high word and C000h in the low word. Numbers returned by GetVelocity must correspondingly be divided by 65,536 to convert to units of counts/cycle. See "SNAP-SCM-MCH16 Conversion Formulas" on page 113.

Restrictions

SetVelocity may not be issued while an axis is in motion with the S-curve profile.

SetVelocity is not valid in Electronic Gear profile mode.

The velocity cannot be negative, except in the VelocityContouring profile mode.

SetVelocity is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

OptoScript Example

```
Status=TransmitReceiveString
(">SetVelocity,0,P1,P2",ComHandle,ResponseString)

Status=TransmitReceiveString (">GetVelocity,0",ComHandle,ResponseString)
```

See Also SetAcceleration, GetAcceleration (page 59), SetDeceleration, GetDeceleration (page 80), SetJerk, GetJerk (page 88), SetPosition, GetPosition (page 95), MultiUpdate (page 55), Update (page 111)

Update

Syntax

Update, *axis*

Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Description

Update causes all buffered data parameters to be copied into the corresponding run-time registers on the specified axis.

The following table shows the buffered commands and variables which are made active as a result of the Update command.

Type	Command
General	ClearPositionError
Trajectory	Acceleration
	Deceleration
	GearRatio
	Jerk
	Position
	ProfileMode
	StopMode
	Velocity
Motor	MotorCommand

OptoScript Example

```
Status=TransmitReceiveString (">Update,1",ComHandle,ResponseString)
```

See Also

MultiUpdate (page 55)

WriteBuffer

Syntax

WriteBuffer, *axis, P1, P2, P3*

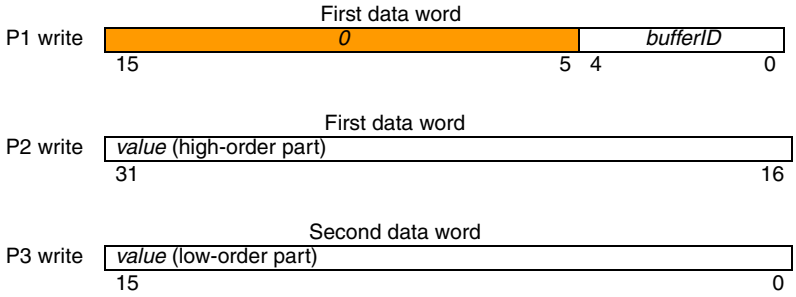
Arguments

	Type	Range	Board Address
axis	unsigned 8 bits	0 to 3	0
		4 to 7	1
		8 to B	2
		C to F	3

Name	Type	Range
bufferID	unsigned 16 bits	0 *to* 31
value	signed 32 bits	-2^{31} *to* $2^{31}-1$

Data Structure

WriteBuffer



Description

WriteBuffer writes the 32-bit value into the location pointed to by the write buffer index in the specified buffer. After the contents have been written, the write index is incremented by 1. If the result is equal to the buffer length (set by SetBufferLength), the index is reset to 0.

OptoScript Example

```
Status=TransmitReceiveString
("">WriteBuffer,2,4,0,3E",ComHandle,ResponseString)
```

See Also

ReadBuffer (page 57), SetBufferWriteIndex, GetBufferWriteIndex (page 78)

SNAP-SCM-MCH16 Conversion Formulas

Use the table on the next page to convert Counts/Cycle, used by some of the motion commands.

	Revolutions/Hour	Revolutions/Minute	Revolutions/Second	Pulses/Second	Pulses/Cycle	Counts/Pulse	Counts/Cycle	% Err
				320 Pulses / Second *	0.000256 Seconds / Cycle *	65536 Counts / Pulse =	5368 Counts / Cycle	-0.013
GetActualVelocity, GetCommandedAcceleration, GetCommandedVelocity, SetAcceleration, GetAcceleration, SetDeceleration, GetDeceleration, SetStartVelocity, GetStartVelocity, SetVelocity, GetVelocity			0.2 Revolutions / Second *	1600 Pulses / Revolution *	0.000256 Seconds / Cycle *	65536 Counts / Pulse =	5368 Counts / Cycle	-0.013
		12 Revolutions / Minute *	1 Minute / 60 Seconds *	1600 Pulses / Revolution *	0.000256 Seconds / Cycle *	65536 Counts / Pulse =	5368 Counts / Cycle	-0.013
	720 Revolutions / Hour *	1 Hour / 60 Minute *	1 Minute / 60 Seconds *	1600 Pulses / Revolution *	0.000256 Seconds / Cycle *	65536 Counts / Pulse =	5368 Counts / Cycle	-0.013
			0.02 Revolutions / Second *	1600 Pulses / Revolution *	0.000256 Seconds / Cycle *	4294967296 Counts / Pulse =	35184372 Counts / Cycle	-2.52E
SetJerk		1.2 Revolutions / Minute *	1 Minute / 60 Seconds *	1600 Pulses / Revolution *	0.000256 Seconds / Cycle *	4294967296 Counts / Pulse =	35184372 Counts / Cycle	-2.52E
	72 Revolutions / Hour *	1 Hour / 60 Minute *	1 Minute / 60 Seconds *	1600 Pulses / Revolution *	0.000256 Seconds / Cycle *	4294967296 Counts / Pulse =	35184372 Counts / Cycle	-2.52E

The number of pluses per revolution depends on your Driver/Amplifier. This is how many pulses are required to complete one revoluion.

Seconds/Cycle is configurable with the SetSampleTime command. Must be divisible by 51.2µS. Minimum value depends on number of axes enabled.